CS396H: Project Descriptions

Fall 2016

Overview

Projects will be carried out in groups of 3 or 4. The final deliverables will be a report and a presentation. The report should be 6-10 pages, consisting of the problem statement, your approach, and your results. The presentation is a summary of the report, and will be given to the rest of the class.

Here is a timeline:

- By 9/21, register your preferences for the various projects listed below by filling out this form: https://docs.google.com/a/cs.umd.edu/forms/d/e/1FAIpQLSe-3M0XtTgIPE8R3YodDagsskHfoWDunwrob8i2mPBoA3hTA/viewform
- By 9/26 I will announce the project groups.
- By 10/12 you should have met with your project mentor to discuss the details of the project, particularly to clarify the goals, approach, timeline, and deliverables.
- By 10/19 write up a detailed 1-2 page plan for the project and share with Prof. Hicks for comment. Begin work!
- By 11/9 submit a 1-page status report, commenting on how you’ve held to the timeline (or not) and any changes in direction.
- By 11/30 submit another status report.
- 12/7, group presentations begin
- Final exam date, group presentations complete, final paper due.

1 A Project on Factoring and Cryptography

Mentor Prof. William Gasarch. It is known that if factoring can be done quickly then Eve can read the message (the converse is not known). Hence factoring is an important problem that people really really want to solve. It is trivial to factor \(N\) in \(O(N^{1/2})\) steps. Pollard’s algorithm improves this to \(O(N^{1/4})\). Or does it? Pollard’s algorithm works well in practice but seems hard to prove works well. Hence we have the usual dilemma: *It works in practice, can we make it work in theory?*

In this project you will

1. Learn and code up Pollard’s algorithm.
2. Run tests to see how fast it runs. Of interest: (i) how big does \(N\) have to be before Pollard is better than the trivial algorithm, (ii) Is it really \(O(N^{1/4})\) in practice? (iii) Find \(c\) such that Pollard’s algorithm works in time \(cN^{1/4}\).
3. Learn and code up RSA.
4. Attack your RSA using Pollard’s algorithm. How large do the parameters of RSA have to be before it cannot be broken by Pollard’s algorithm?

Some relevant papers/slides:
- [https://en.wikipedia.org/wiki/Pollard%27s_rho_algorithm](https://en.wikipedia.org/wiki/Pollard%27s_rho_algorithm)

## 2 Finding New Splicing Events in RNA-seq Data

Mentor **Prof. Zia Khan**. The vast majority of human genes encode mRNAs that are translated into proteins to perform functions within and outside of our cells. A gene is transcribed\(^1\) to produce a pre-mRNA that undergoes a process called splicing,\(^2\) where regions of the pre-mRNA transcripts called introns are removed leaving a mature mRNA transcript. This process is quite complex and a cell can often select which introns to remove by a process called alternative splicing.\(^3\) It is thought that the presence of introns allows our genes to be regulated more precisely, and the use of alternative splicing allows our cells to produce a complex repertoire of proteins, called isoforms, using a limited subset of genes. Because this process is so important, defects in splicing are known to cause and contribute to many terrible human diseases including cancer.

The extent of alternative splicing of human genes is still unknown. One of the technical challenges of identifying isoforms is that the most sensitive technology for detecting rare isoforms can only produce very short reads\(^4\) and these reads will be split across introns that were removed from a mature mRNA.

The goal of this project is to develop an algorithm that can distinguish alternatively used introns from short read RNA-seq data that have been previously documented from those that have not been previously seen. The challenge will be to understand noise in these data and develop an approach that is tunable and robust as these data have substantial noise. You should implement your algorithm using Python. You should use a combination of tools including pysam\(^5\) and interval tree.\(^6\). Your standard for what is “known” should be the human gene annotations from GENCODE.\(^7\)

You will be given .BAM alignment files from RNA-seq data was collected from several classes of immune cells, including T-cells and B-cells. The research questions you will try to answer follow: Are there clusters of alternatively used introns in these classes of immune cells that have not been previously seen? How confident are you in these discoveries? Does there exist an algorithm for which you can quantify the sensitivity and the specificity with which new introns are discovered?

You can evaluate your algorithm by “removing” known intron clusters from GENCODE annotations and seeing how well you recover these removed annotations. You are encouraged to do so as well as report any new discoveries in the data provided.

Notes:
6. [https://pypi.python.org/pypi/intervaltree/2.0.4](https://pypi.python.org/pypi/intervaltree/2.0.4)
7. [http://www.gencodegenes.org](http://www.gencodegenes.org)

## 3 Inferring Type Signatures for Ruby Methods with Machine Learning

Mentor **Prof. Jeff Foster**. In class Jeff will talk about RDL, a new type system we’ve been developing for Ruby. (RDL is called “Hummingbird” in the paper you’ll read.\(^1\)) To use RDL, developers annotate their methods with type signatures, and then RDL checks whether method bodies match their annotated types. RDL is designed so developers can gradually add type signatures to their programs, but it still requires a fair amount of effort to come up with those signatures. This is particularly challenging because there could be many possible valid type signatures for the same method. For example, is a method meant to accept only Fixnums or can it accept arbitrary numbers? Can a method accept any object with a to_s method or can it only accept Strings and Symbols?
Jeff’s research group has tried a few approaches to solving this problem in the past, but none has really worked well. The goal of this project would be to explore a new idea: can machine learning be used to discover good type signatures for Ruby methods?

More specifically, the input to the machine learning (ML) algorithm could be any or all of the following: the method name, argument names, observations of run-time types of arguments from test cases, and the syntactic structure of the method body (or more!). We’ll probably use the Keras\(^2\) deep learning library. Since Jeff is not an expert on ML, you’ll also get some advice from Prof. Hal Daum III and one of his grad students.

The project will be to develop a prototype ML tool for this purpose, evaluate its effectiveness, and, if it seems promising and time permits, make iterative improvements to it.

(Aside: Jeff is also looking to hire some students to do (more straightforward) web programming for a Ruby on Rails application he is building; please ping him if you’re interested in that. The web app project will use RDL, too, but that will not be its main focus.)

Notes:
\(^2\) https://keras.io

4 Evaluating defenses against Return Oriented Programming (ROP) attacks

Mentor Dr. Nick L. Petroni, Jr. In class we’ll discuss Return-Oriented Programming (ROP), an exploitation technique that cleverly reuses existing code in a target program to execute arbitrary attacker logic. Since the development of ROP (and return-to-libc before it), several defensive techniques have been proposed, prototyped, and deployed in mainstream software. Similarly, several advancements in attacker techniques, including defeats for the latest defenses, have also been published. The arms race continues.

Whichever of the three projects below you might do, the first step is to briefly survey the latest advances in anti-exploitation techniques. In your final paper, briefly describe what has been proposed and which can we find in commodity software today. Similarly, identify the latest in offensive thinking. In both cases, you may choose to narrow your survey to a specific class of vulnerabilities, e.g., memory corruption (stack, heap, or both) or based on some other criteria. Be clear about your scope.

Project 1: Choose a particular exploitation defense and describe (or, even better, demonstrate) how it would prevent the exploitation of one or more specific vulnerabilities (make sure it’s not one already used in a paper example!). A good start would be to survey the Common Vulnerabilities and Exposures (CVE) list\(^1\) or look at proof-of-concept exploits in the public domain (e.g., as part of Metasploit\(^2\)).

Project 2: Propose a new anti-exploitation technique or an improvement to one that you found in your survey. Explain how your approach compares with existing system(s). Offer some theories about the strengths and weaknesses of your proposal. Validate at least some part of your technique either empirically (through direct implementation or simulation) or analysis — what is the extent of the protection it can provide?

Project 3: Assess qualitatively and validate experimentally an already-published weakness in a recent anti-exploitation system. For example, in “Evaluating the Effectiveness of Current Anti-ROP Defenses” (RAID 2014)\(^3\) Schuster et al. describe problems with kBouncer, ROPecker, and ROPGuard. Pick one of those systems and reproduce their findings enough to convince yourself whether they are correct. Can you think of any changes that would improve the chosen defensive technique?

Notes:
\(^1\) https://cve.mitre.org/cve/
\(^2\) https://www.metasploit.com/
\(^3\) http://hgi.ruhr-uni-bochum.de/media/emma/veroeffentlichungen/2014/07/29/antiROP-RAID14.pdf
5 Automated Testing

Mentor Prof. Michael Hicks. The Driller paper\(^1\) (which we will read in class) discusses two types of automated testing: \textit{fuzz testing} and \textit{concolic execution}. The former is exemplified by AFL\(^2\) and the latter by KLEE.\(^3\) The claim of the paper is that the two approaches have complementary strengths, and that a tool combining the two brings out the best of both.

For this project, your job is to run your own experiment comparing KLEE and AFL. You will pick the benchmark programs to run on, and set the experimental parameters for comparing the two. Think carefully about your choices and how they effect the meaning of the results. In the end you want to answer the question: Do you believe the claims of the Driller paper that (a) the two approaches are complementary and (b) if so, that this is the case for the reasons stated (or, if not, the reasons for your differing opinion)\(^?\)

Notes:
\(^1\) \url{https://www.internetsociety.org/sites/default/files/blogs-media/driller-augmenting-fuzzing-through-selective-symbolic-execution.pdf}
\(^2\) \url{http://lcamtuf.coredump.cx/afl/}
\(^3\) \url{https://klee.github.io/}

6 Test equivalence in bug checking

Mentor: Prof. Michael Hicks. When do two tests identify the same bug? As per a blog post I wrote some time back,\(^1\) this question is not as obvious as it might seem. The projects here aim to explore this question in different ways.

6.1 Project: When are two test cases “the same”?\(^1\)

Can we automatically determine whether two failing test cases T1 and T2 that test a program P identify the same bug in that program? “Same” is fungible, but one potential definition is that T1 and T2 identify the same bug if a “small” fix of P causes both T1 and T2 to pass. As such, two test cases being “the same” is not an absolute property, across all programs, but is a property relative to a particular program. A reasonable alternative problem to explore is to figure out if a fix to P fixes more than one distinct bug, and thus incorrectly “unifies” more than one test case.\(^1\)

Ideally, this determination would be: (1) Language-independent (or mostly so), (2) Quantifiably reliable, (3) Efficient – But its OK to only get some of these things, as long as there is progress. As additional information a solution could use, we can provide a variety of valid test cases. And we could provide a variety of implementations.

There are multiple ways one might try to solve this problem, and if you are interested in this, we can discuss some of them. One potential approach would be to analyze the artifacts of execution (e.g., call stack coverage).

6.2 Project: What is a bug (empirically speaking)?\(^2\)

In order to do the project above, we need some reasonable definition for what a bug is. One idea is to use empirical analysis to figure this out. In particular, we could analyze issue reports and resolutions (bugfixes) on Github. Would most fixes be localized into a single code unit, supporting the idea that a bug tends to be local to a particular code unit? How often would we see duplicate or overlapping bug reports, suggesting that an initial report really involved multiple bugs, and not a single bug?

This project would be to do an empirical analysis of issue reports for some set of projects on GitHub, and to use that analysis as a basis for a refined definition of what constitutes a (single) bug. As background, you might check out papers that systematically analyze GitHub data.\(^2\)

Notes:
\(^1\) Blog post on “What is a bug?” \url{http://www.pl-enthusiast.net/2015/09/08/what-is-a-bug/}
\(^2\) See the GitHub archive for data to analyze (\url{https://www.githubarchive.org/}), and what researchers have done with it (e.g., \url{http://macbeth.cs.ucdavis.edu/odd-bugs.pdf}, and papers it cites)
Organ Exchange

Mentor Prof. John Dickerson. The preferred treatment for kidney failure is a transplant; however, demand for donor kidneys far outstrips supply. Kidney exchange, an innovation where willing but incompatible patient-donor pairs can exchange organs—via barter cycles and altruist-initiated chains—provides a life-saving alternative. Patients who are fortunate enough to find a willing living donor must still contend with compatibility issues like blood type, tissue type, and other medical or logistical factors (as we discuss later in the paper). If a willing would-be donor is incompatible with a patient, the kidney cannot be transplanted.

Kidney exchange is a recent innovation that allows patients to swap willing but incompatible donors. Figure 1 shows a graphical representation of a small pool consisting of three patient-donor pairs, where an arrow from pair $i$ to pair $j$ means the patient at $j$ is compatible with the donor at $i$. Also shown is an altruistic donor; such donors do not come with paired patients and are willing to donate a kidney without asking for one in return. The basic kidney exchange problem is then to recommend an optimal—according to some social welfare function—set of disjoint cycles and altruist-initiated chains in the graph.

![Figure 1: Tiny example kidney exchange pool with three patient-donor pairs and one altruistic donor.](image)

7.1 Project: Learning QALYs

Not all kidneys are created equal. Indeed, some kidneys are worse than others globally, while other kidneys might be a better match for me than they are for you. One measure of this is graft survival rate, the length of time a donor organ continues to work in its recipient before requiring replacement. This, along with the recipient’s health at the time of transplant, gives an estimate of how many “QALY”s a kidney is worth to a potential recipient. QALYs—quality-adjusted life-years—are a generic measure of disease burden, including both the quality and the quantity of life lived. It is used in economic evaluation to assess the value for money of medical interventions. One QALY equates to one year in perfect health. The more QALYs, the better.

In this project, you will use machine learning to estimate QALYs for potential donor organs and recipients. You will use a large dataset covering all live kidney donations in the US since 1987—roughly 100000 or so—to learn a classifier that maps potential donor organs to a QALY score. Once that classifier is learned, you will add this classifier to a large kidney exchange codebase used to do sensitivity analysis at the UNOS kidney exchange. How much does optimizing with respect to QALYs help relative to the status quo?

Notes:
- OPTN data: [https://optn.transplant.hrsa.gov/data/request-data/](https://optn.transplant.hrsa.gov/data/request-data/)
- Codebase: [https://github.com/JohnDickerson/KidneyExchange](https://github.com/JohnDickerson/KidneyExchange)

7.2 Project: Greedy Packing

Finding the maximum cardinality disjoint set of small cycles and long chains in a directed graph is NP-hard. Lots of research has gone into solving this problem to optimality in practice. Still, solving the problem is quite slow. There is value in finding approximate solutions—that is, creating algorithms that quickly find a good, but maybe not the best,
packing. These algorithms could be deployed in dynamic optimization packages that are only now starting to be used in real exchanges.

In this project, you will design heuristic methods for packing cycles and chains in real kidney exchange graphs. You will compare them against some state-of-the-art optimal packing algorithms. How close do you get to optimal, and with what variance? How much time do you save versus optimality? Can you prove anything about your heuristics?

Notes:
Optimal packer paper: https://arxiv.org/abs/1606.01623
Code: https://github.com/jamestrimble/kidney_solver

8 Prediction Markets

Mentor: Prof. John Dickerson. In a prediction market, participants trade based on their belief that some binary event in the future will occur (e.g., a specific political candidate wins/loses an election, or a specific sports team wins/loses a game). Participants buy shares at a price which fluctuates between $0.0 and $1.0 according to market forces. If the event occurs (e.g., the candidate wins, the team wins), all participants receive $1.0 per share held; if the event does not occur (e.g., the candidate loses, the team loses), participants receive $0.0.

If a participant believes an event will occur with probability \( p \in [0, 1] \), and the current market price is \( p' > p \), then she will sell her shares (because her expected profit by holding, \( p \), is less than her expected profit from selling, \( p' > p \)). Similarly, if the price is \( p' < p \), then she will buy shares, because her expected profit from buying a share is greater than the price spent to acquire that share. In this way, prediction markets aggregate information from disparate sources, implementing a “wisdom of the crowds” approach to predicting whether or not an event will occur.

You will build a prediction market for an event of your choosing. The event should have informative clues happen during the semester. Steps:

1. Read “The Gates-Hillman Prediction Market,” a paper about the implementation and performance of a market used at CMU in 2009 to predict the opening of their Gates-Hillman center;
2. Design your prediction market (including the pricing mechanism, how initial funny money is distributed, etc);
3. Implement the market mechanism and design a website where buyers and sellers can enter into trade; and
4. Watch as people trade! Is there actually wisdom in the crowd?

Notes:
Predictwise: http://predictwise.com/
GHPM paper: https://www.cs.cmu.edu/~aothman/ghpm_journal.pdf

Strongly encouraged: Whatever prediction market you build, also use it to predict the opening of the Brendan Iribe Center (http://iribe.cs.umd.edu/), the UMD CS Department’s new building, which is tentatively scheduled to open Fall 2018. Prediction markets have been used to try to more accurately predict the actual move-in date for new CS buildings (for example, at CMU in 2009 and at UT Austin in 2014). The website you put up this semester should stay live until the opening of the new CS building.

9 Twitter Bots

Mentor: Prof. John Dickerson. The openness of Twitter’s platform allows for, and even promotes, programs that automatically post. These “bots” post content ranging from helpful (e.g., recent news stories or public service announcements) to malicious (e.g., spam or phishing links). Such malicious bots are alleged to help political candidates skew public perception; for instance, botornet.net asserts that former US Presidential candidate Newt Gingrich gained over a million followers on Twitter through the use of bots, a charge Mr. Gingrich is reported to have denied.
Similarly, during last election cycle, Hill reported that “up to 29.9% of Barack Obama’s 17.82 million [Twitter] followers and 21.9% of Mitt Romney’s 814,000 followers may be fake”. In short, there is now a widespread belief that bots constitute a significant part of the social media world—and that many of them are malicious in their intent.

In this project, you will build a classifier that determines which users on Twitter (or a similar social network) are bots. You can use whatever features you’d like—sentiment, text saliency, temporal elements, social graph structure, etc—to do this. We will create a “gold standard” data set via Mechanical Turk or similar to properly train/validate/test your classifier.

Notes:

10 Roommate Swaps
Mentor: Prof. John Dickerson. UMD ResLife has a “room exchange,” wherein undergrads (aka you!) in on-campus housing can swap roommates during some predefined period mid-semester. See: http://reslife.umd.edu/housing/reassignments/roomexchange/

The way this exchange seems to be executed is fairly naive, allowing:

- only pairwise swaps (that is, given two rooms A and B with roommates A1/A2 in A and B1/B2 in B, A and B can participate in a swap to achieve A1/B2 and A2/B1—but, given three rooms A, B, and C, a swap such that e.g. A1/B1 A2/C1 and B2/C2 would not be allowed);
- fairly weak expression of preferences over alternate roommates (that is, it sounds like roommates have to find each other in an ad-hoc fashion rather than expressing features over “ideal roommates” and then being reassigned); and
- the swaps seem to only occur during one pre-defined period each semester.

Interestingly, moving from 2-swaps to 3-swaps or more makes the problem NP-hard instead of PTIME to compute (for pretty much any reasonable definition of ‘the problem’; we can talk about this sometime), but one would expect utility gains w.r.t. the participants, so there are some interesting optimization problems to be tackled here.

Similarly, it’d be cool to make a general “room swap” website that we could first deploy at UMD and later deploy at other universities. On-campus housing allocation is essentially identical at most schools, so this could have high impact!

For this project, you will:

- Formalize the room exchange currently implemented by Maryland;
- Determine if anything is left “on the table” by the simplicity of the current formulation;
- Augment the formulation to include these useful additions; and
- Implement the swap mechanism in a website.

11 Measurement of the code signing ecosystem
Mentor Prof. Tudor Dumitras. When operating systems decide what software can be trusted to execute, they base this decision on digital signatures associated with the executables and X.509 certificates that allow them to identify the organizations to which the digital signatures belong. Measurement studies of SSL certificates in the wild have illustrated interesting vulnerabilities in this ecosystem, such as a large number of active certification authorities, incorrect certificate chains, or practical challenges for revoking compromised certificates. The code signing ecosystem has not been measured in the same way and could potentially yield equally interesting insights.
You will be given access to a data set of executables observed on 5 million Windows hosts and the corresponding certificates. The research question in this project is to map the current trust decisions which organizations are we really trusting when we run code on our computers. For example, can you identify threats against this trust model, similar to the ones found in the SSL certificate ecosystem? Which certificate authorities sign malware or potentially unwanted software? Can you identify certificate polymorphism (one organization signing executables using multiple certificates, with different names)? Can you identify legitimate certificates that have been compromised and used for malicious purposes?

Some pointers on code signing:


12 Build it, Break it, the Language Edition

Mentor, Hal Daumé III.

In a new research project (joint with Emily M. Bender at University of Washington), we wish to explore the intersection of natural language processing (NLP), computational linguistics and machine learning. Our main goal is to help researchers better understand errors that their NLP systems are making, by providing a platform for stress-testing against adversarial inputs. We will accomplish this in a similar way to the “Built it, Break it, Fix it” (https://builditbreakit.org) contest that other researchers at UMD have been running recently in the cybersecurity domain. In the language setting, builders will build systems for solving some particular natural language processing problem (e.g., machine translation from Mandarin to English); then breakers will attempt to provide Mandarin sentences as input to these systems that cause them to produce incorrect output. The goal is to learn more about what our current technology is good (and bad) at through this process.

There are two major components of this project. The first is to build an online system that will enable builders and breakers to interact. This is being handled external to this 396H project. The second component, which is the focus of this 396H project is to develop (a) an evaluation scheme for the submitted “breaker” test examples and (b) build one or two of our own “builder” systems to submit to the contest. For (a), we will use crowdsources (e.g., mechanical turk or crowdflower) to construct validation tasks for the chosen natural language processing problem, possibly paired with active learning strategies to ensure builders and breakers are behaving honestly. For (b), we will use a combination of open source NLP tools and machine learning tools to make initial solutions to the “builder” problem.