SAND2019-4180R

Exploiting BWA to Manipulate Raw Genomic Data In-Place

PRESENTED BY

Corey Hudson

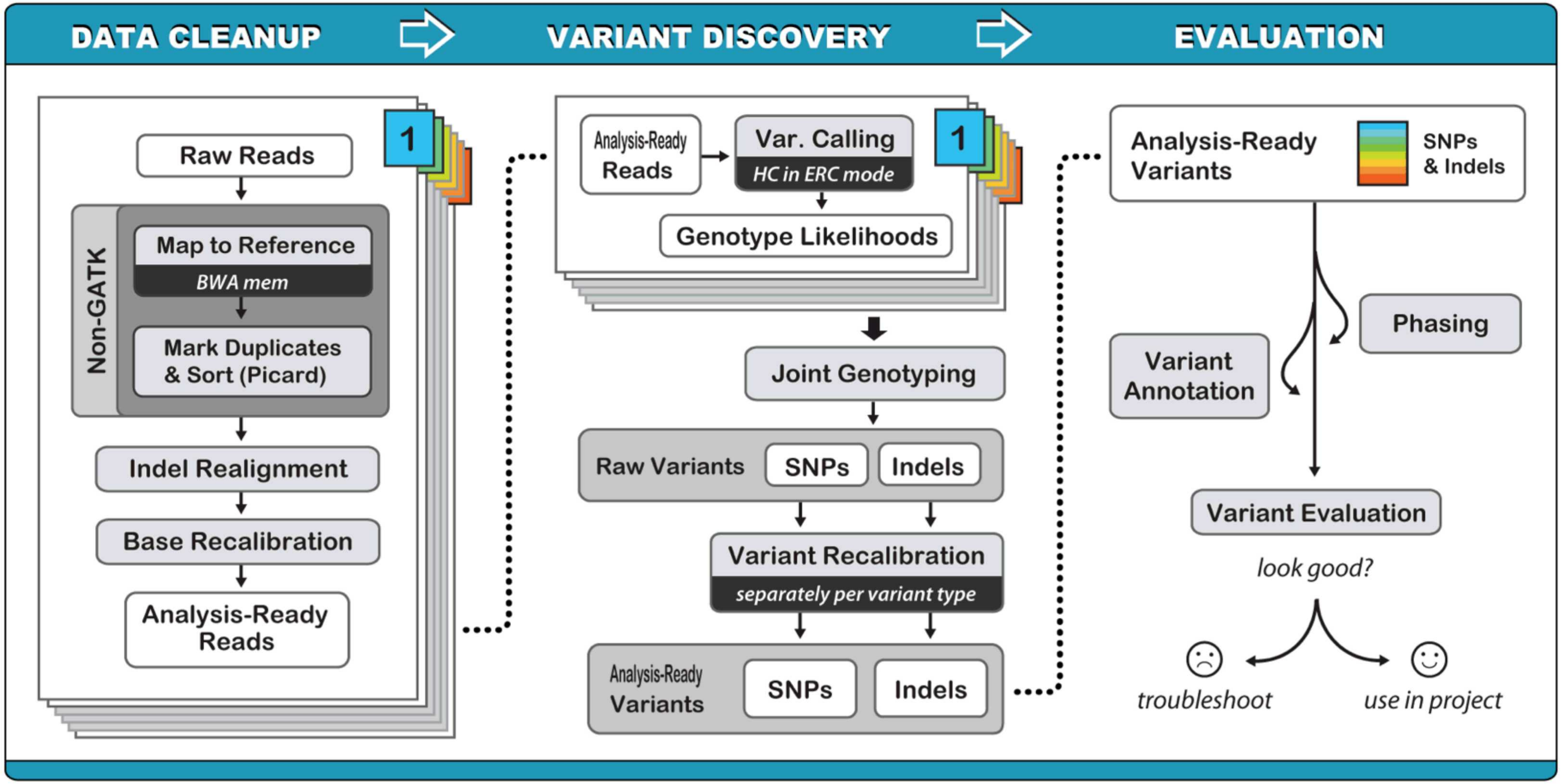
Sandia National Laboratories

Exploring implications of genomic software vulnerabilities under realistic security assumptions

Goals for modeling genomic security

1. *Only allow **standard** simplifying security assumptions*
   a) ASLR/DEP, secure compiling off – fairly standard and can be bypassed, but at the cost of clarity of exploit
   b) Adversarial presence on the network – assumes breach is possible
2. *Use existing knowledge of genomics systems to frame assumptions*
   a) Allow read/write of raw-data – realistic based on experience in genomics
   b) Mixing of databases and executable software – also realistic based on experience in genomics
   c) Download databases using the standard protocols
   d) Any authentication must use the standard protocols
3. *Use a vulnerability that exists* in the wild
4. *Use a standard best-practices genomics pipeline*
5. *Manipulate raw data, prior to analysis and complete analysis without issuing errors*

Standard Best-Practices for Genomic Variant Detection

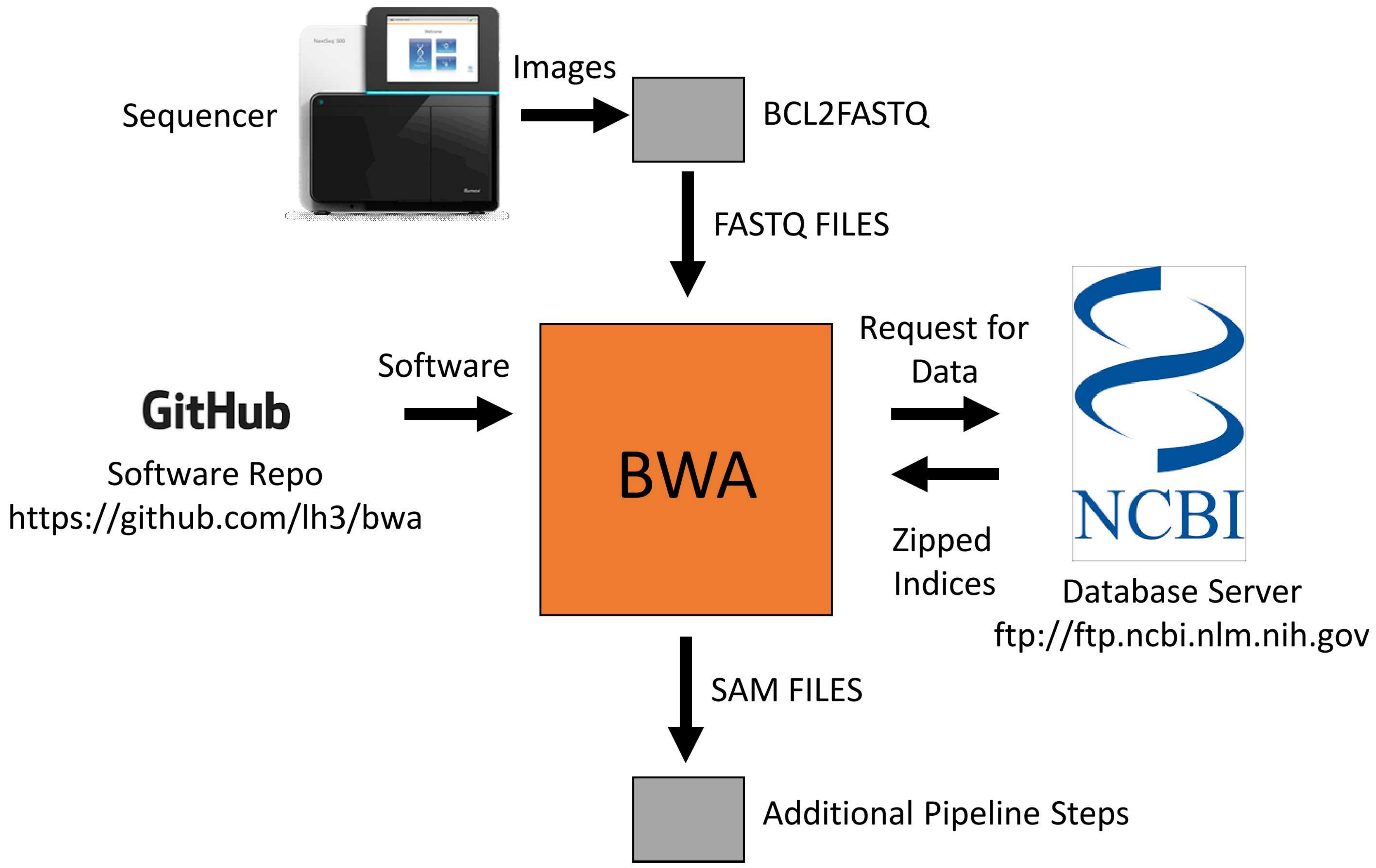


Best Practices Pipeline for Variant Detection, per BROAD Institute:
https://gatkforums.broadinstitute.org/gatk/discussion/3238/best-practices-for-variant-discovery-in-dnaseq

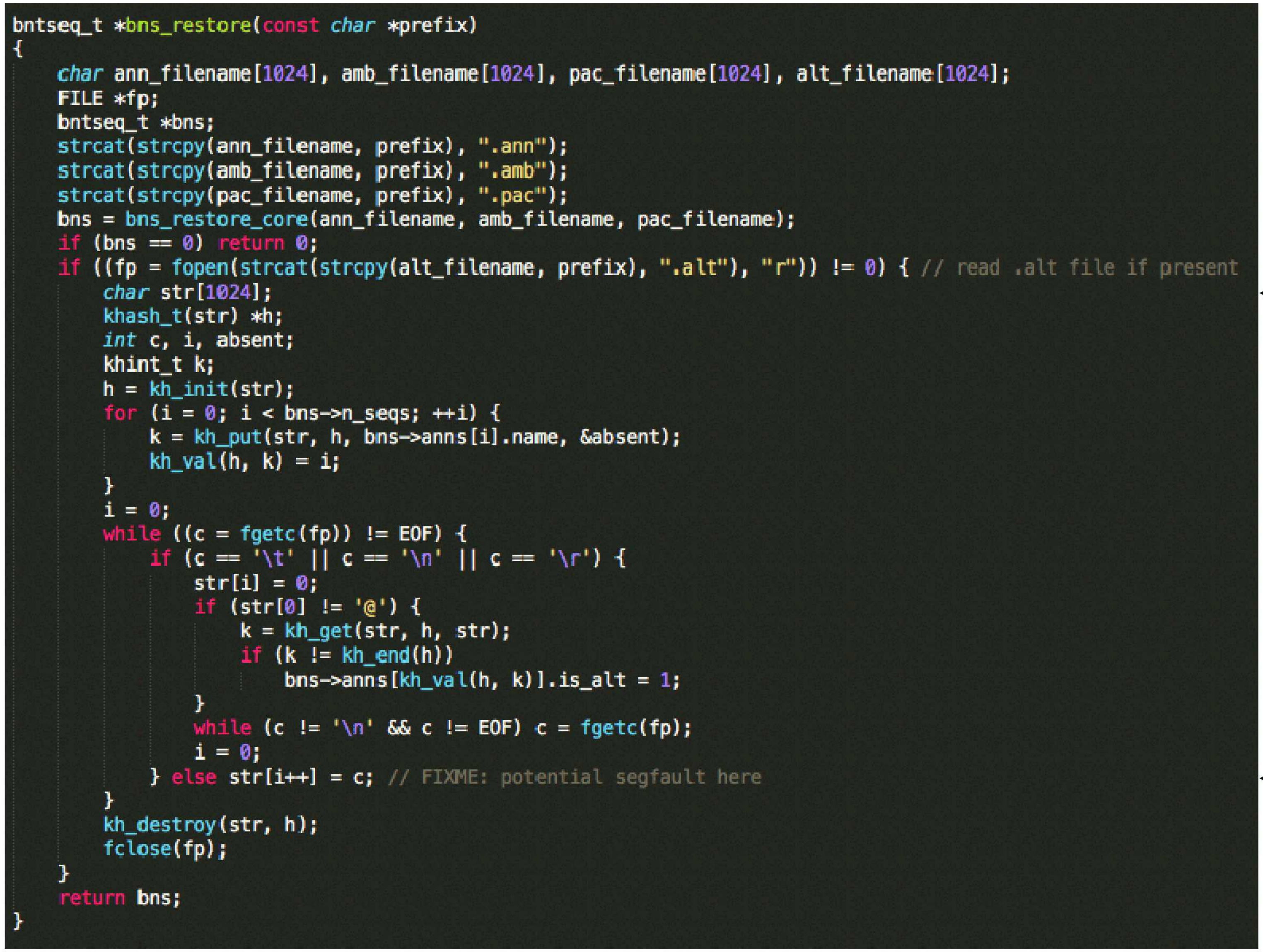
First piece of software in best-practices pipeline is BWA

1. BWA takes FASTQ files as input and maps these to a reference genome, creating a SAM file
2. In 2014, BWA developers added the ALT-aware capacity – which allowed users to map reads to a population, rather than canonical single reference
3. Since the population is always changing and requires up-to-date knowledge, the reference is hosted at a central repository
4. BWA provides a tool – bwa.kit, which accesses this data from the US National Center for Biotechnology Information (NCBI), which has provided resources for the storage and delivery of these files as a tarred and gzipped directory of indices: ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/001/405/GCA_000001405.15_GRCh38/seqs_for_alignment_pipelines.ucsc_ids/
5. The user then unzips and stores the indices provided by NCBI
6. A **.alt** file is used to index the genome and make it alt-aware

BWA interactions with outside data



BWA has a vulnerability in its native codebase



```c
bntseq_t *bns_restore(const char *prefix)
{
    char ann_filename[1024], amb_filename[1024], pac_filename[1024], alt_filename[1024];
    FILE *fp;
    bntseq_t *bns;
    strcat(strcpy(ann_filename, prefix), ".ann");
    strcat(strcpy(amb_filename, prefix), ".amb");
    strcat(strcpy(pac_filename, prefix), ".pac");
    bns = bns_restore_core(ann_filename, amb_filename, pac_filename);
    if (bns == 0) return 0;
    if ((fp = fopen(strcat(strcpy(alt_filename, prefix), ".alt"), "r")) != 0) { // read .alt file if present
        char str[1024];
        khash_t(str) *h;
        int c, i, absent;
        khint_t k;
        h = kh_init(str);
        for (i = 0; i < bns->n_seqs; ++i) {
            k = kh_put(str, h, bns->anns[i].name, &absent);
            kh_val(h, k) = i;
        }
        i = 0;
        while ((c = fgetc(fp)) != EOF) {
            if (c == '\t' || c == '\n' || c == '\r') {
                str[i] = 0;
                if (str[0] != '@') {
                    k = kh_get(str, h, str);
                    if (k != kh_end(h))
                        bns->anns[kh_val(h, k)].is_alt = 1;
                }
                while (c != '\n' && c != EOF) c = fgetc(fp);
                i = 0;
            } else str[i++] = c; // FIXME: potential segfault here
        }
        kh_destroy(str, h);
        fclose(fp);
    }
    return bns;
}
```

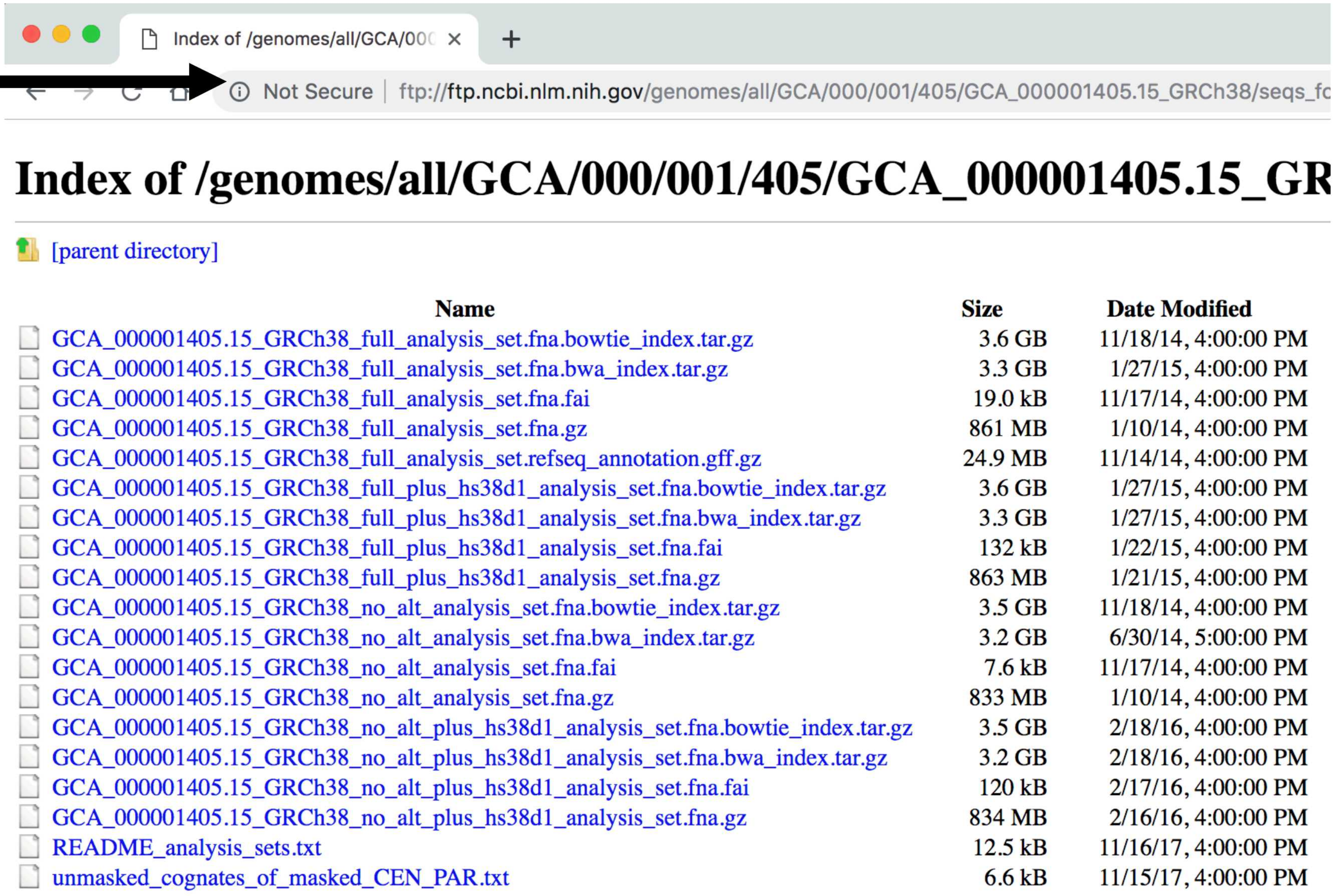
1024 byte buffer

If a .alt file has a line >1024 bytes it will overflow here

.ALT files are delivered over unencrypted channels

FTP Protocol

No checksums to validate data transfer



Index of /genomes/all/GCA/000

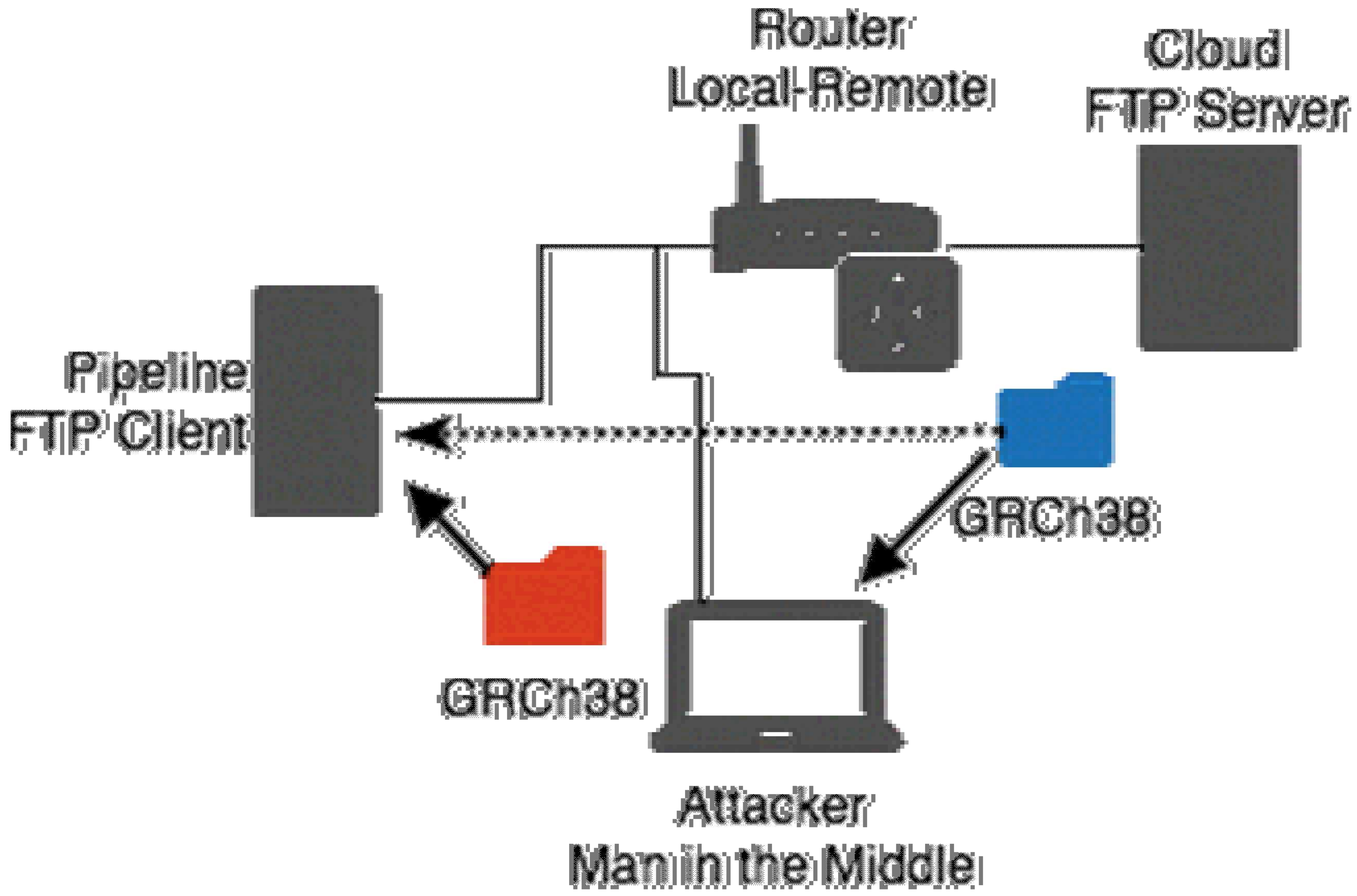
Not Secure | ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/001/405/GCA_000001405.15_GRCh38/seqs_fo

Index of /genomes/all/GCA/000/001/405/GCA_000001405.15_GR

[parent directory]

Name	Size	Date Modified
GCA_000001405.15_GRCh38_full_analysis_set.fna.bowtie_index.tar.gz	3.6 GB	11/18/14, 4:00:00 PM
GCA_000001405.15_GRCh38_full_analysis_set.fna.bwa_index.tar.gz	3.3 GB	1/27/15, 4:00:00 PM
GCA_000001405.15_GRCh38_full_analysis_set.fna.fai	19.0 kB	11/17/14, 4:00:00 PM
GCA_000001405.15_GRCh38_full_analysis_set.fna.gz	861 MB	1/10/14, 4:00:00 PM
GCA_000001405.15_GRCh38_full_analysis_set.refseq_annotation.gff.gz	24.9 MB	11/14/14, 4:00:00 PM
GCA_000001405.15_GRCh38_full_plus_hs38d1_analysis_set.fna.bowtie_index.tar.gz	3.6 GB	1/27/15, 4:00:00 PM
GCA_000001405.15_GRCh38_full_plus_hs38d1_analysis_set.fna.bwa_index.tar.gz	3.3 GB	1/27/15, 4:00:00 PM
GCA_000001405.15_GRCh38_full_plus_hs38d1_analysis_set.fna.fai	132 kB	1/22/15, 4:00:00 PM
GCA_000001405.15_GRCh38_full_plus_hs38d1_analysis_set.fna.gz	863 MB	1/21/15, 4:00:00 PM
GCA_000001405.15_GRCh38_no_alt_analysis_set.fna.bowtie_index.tar.gz	3.5 GB	11/18/14, 4:00:00 PM
GCA_000001405.15_GRCh38_no_alt_analysis_set.fna.bwa_index.tar.gz	3.2 GB	6/30/14, 5:00:00 PM
GCA_000001405.15_GRCh38_no_alt_analysis_set.fna.fai	7.6 kB	11/17/14, 4:00:00 PM
GCA_000001405.15_GRCh38_no_alt_analysis_set.fna.gz	833 MB	1/10/14, 4:00:00 PM
GCA_000001405.15_GRCh38_no_alt_plus_hs38d1_analysis_set.fna.bowtie_index.tar.gz	3.5 GB	2/18/16, 4:00:00 PM
GCA_000001405.15_GRCh38_no_alt_plus_hs38d1_analysis_set.fna.bwa_index.tar.gz	3.2 GB	2/18/16, 4:00:00 PM
GCA_000001405.15_GRCh38_no_alt_plus_hs38d1_analysis_set.fna.fai	120 kB	2/17/16, 4:00:00 PM
GCA_000001405.15_GRCh38_no_alt_plus_hs38d1_analysis_set.fna.gz	834 MB	2/16/16, 4:00:00 PM
README_analysis_sets.txt	12.5 kB	11/16/17, 4:00:00 PM
unmasked_cognates_of_masked_CEN_PAR.txt	6.6 kB	11/15/17, 4:00:00 PM

Modelling Network Vulnerability Using Emulytics



Steps in Attack

Setup

1. Get presence of host network
2. Spoof FTP data transfer
3. Have remote machine stop database transfer from NCBI and deliver poisoned .ALT file
4. When BWA reads poisoned .ALT file it will trigger a buffer overflow
5. Use overflow to issue a command to overwrite all .FASTQ files in the system to change one sequence, to another sequence and complete analysis

Outcome

1. Continue to process files using standard workflow
2. Result will be a different genotype for all files in the system
3. Final .vcf files (standard genotype format) will report new genotypes

Proof of Concept: In-Place Data Manipulation

1. Search for unique sequence in all .fastq files:

`CACAGAAAGCTAATGGG`

2. Replace with new sequence differing by one character:

`CACAGAACGCTAATGGG`

3. Empirical Result in VCF for all files:

- Statistically significant difference between files – with and without exploit
- **Without exploit** – Genotype **AA** at chromosome 12 position 64544989
- **With exploit** – Genotype **AC** ($P<10^{-200}$) at same position

What did this exploit prove?

Using a realistic set of security assumptions it is possible to use a vulnerability that exists in the wild to change the outcome of genomics tests by manipulating raw data and have the exploit trigger no errors

What can be done?

- We worked with BWA to issue a patch that invalidates the exploit https://github.com/lh3/bwa/pull/232
- We are working with US-CERT/DHS to publicize issue to encourage updating and blogging the issue
- **Cyber-hygiene issues**: Do not allow raw files to be write-enabled, encourage websites to encrypt or provide checksums

Security Assumptions in DNA-Encoded Malware Exploit

1. The exploited software (FQZCOMP) in that exploit is not part of standard best-practices for preprocessing variant data

2. FQZCOMP is unnecessary and its functions can be performed by standard Unix/Linux/MacOSX/Windows commands, for example `gzip *.fq`

3. A vulnerability must be added to the main code repo – seldom an accepted security assumption

4. Current sequencing technology is ~300 bp at most in length of individual reads
   - A. Running on a 32-bit machine, overflow requires ~380 bp sequence to issue even a simple command and these must have no errors, not be reversed and be complete in length
   - B. Running on 64-bit machine, overflow would require ~760 bp sequence with the same issues

5. The FASTQ file going in to the exploited FQZCOMP would have to be much different than what is currently standard

6. Any sequencing errors can invalidate an exploit that required a synthetic gene to be inserted into the sequenced organism