Malware Analysis / Reverse Engineering

Bob Nissen

Bob

- 42 Years with NSA
- Overseas Field Tour
- Last 20 in Cyber Security
  - Subject Matter expert (SME) in Malware Analysis
  - Analyze malicious code employing static analysis, reverse engineering and dynamic analysis techniques.
  - Reverse engineer communications protocols
  - Determine initial infection and capabilities of the malware

Why Do Malware Analysis?
Find / Stop

- Quick Triage v In-depth analysis
  - Time available
  - Focus of Analysis
- Find
  - Develop Indicators of Compromise
  - Infected systems
  - Network traffic
- Stop
  - Removal
  - Neuter
  - Block or re-direct

Why Do Malware Analysis?
Learn

- Adversary (Developer and User)
  - Interests
  - Attribution
    - Difficult
  - Which features are under whose control?
- Capabilities
    - Data collect or destroy
- Interoperable?
  - Common code base or design base?
  - Across OS’s

Why Do Malware Analysis? Understand

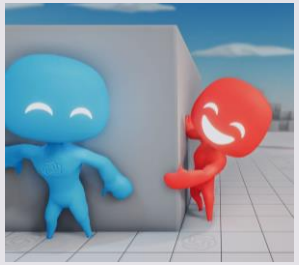
- MW Command and Control
  - Emulate
  - Decode exfiltration
- Damage assessment
- Lineage
  - not always more, refactoring can be less
- Archive helps

Why Do Malware Analysis? Understand

- Assessing programmers' skill
  - Programmers background
  - Sometimes looks like a class assignment
- Concentrate on what we know
  - IP address - bunch of ASCII digits or a 32 bit hex number?
  - Three XOR's in a row
- Code can hide, but its has to run
  - Obfuscation can't get in way of operations
  - Obfuscation as a signature
- Design is not "yours"

Malware Reverse Engineering
Some This's and That's

- Can hide, but has to run
  - Assume it "works"
  - Design is "good"
  - Not how I would have written it
    - Code made sense to 'someone'
- Sudoku
  - Need to zoom in and out
  - Iterative
- Balance between static and dynamic
  - Difficulty in jump jamming



3	4			7		2	9	
7			3					
	9		2		6		4	
	8	7	9		5		3	2
		3		2		6		
		4			3		1	
1	6		5	9		3		
			1	3	2	8		5
	3	8			4		2	9

Malware Reverse Engineering Some This's and That's

- Code v API's
  - Nothing happens at the system level without API's
  - Nothing happens at the bit level without code
- Careful with high level code, especially decompilers
- It's the machine code, not the comments
  - Complex processors, can you even trust assembly?
  - Reverse Engineering x86 Processor Microcode CanSecWest 2018
- Objects recovery
  - Under the hood, 'just' structures
  - Mapping methods to objects
    - Can look like "unexplored" code

Malware Reverse Engineering
Some This's and That's

- Programmer or compiler?
  - Beware compiler optimization
    - Optimizing out zeroing an array just before freeing
    - Unrolling loops
- Wrapper (marshaling) look-alikes
  - Read/write get wrapped almost the same
  - Difference that matters?
- Beware of printf and other time sucks
  - 'Right of passage'
- Understanding different parameter passing conventions

Perils of Dynamic Execution Miscellaneous Thoughts

- Long timeouts
  - Long loops w/unused results
  - MW running out the clock
- Discontinuous API calls
  - Data flow/tracking
- Multiple Threads
  - Always watching each other
  - Emulation, stepping: How/when to switch
- Treatment of Dropped Files
  - Once get second generation, how to ‘source’
- Timeliness and Source DNS for ‘Live’
  - Under adversary’s control

Perils of Dynamic Execution Miscellaneous Thoughts

- Deniable Encryption
  - Desired Text xor D-Key ➔ Encrypted Text
  - Encrypted Text xor Alternate Text ➔ A-Key
  - Transmit (and assume interception of) Encrypted Text
  - Make A-Key “easy” to find
  - A-Key xor Encrypted Text ➔ Alternate Text
- Revert and OS State
  - Saving User space v kernel v disk
- Thread Local Storage
  - TLS initialization callback runs *before* entry point.

Perils of Dynamic Execution
Consider the Source

In Transit

- Could be Self Contained
- May download additional components
- Sense Environment
- May drop additional files
- May write registry settings

In Stalled

- Environmental Dependencies
  - Reliant on other pieces
    - Registry settings
    - Configuration files
  - OS dependencies
    - DLL’s
  - ‘Live’ in a certain path

Perils of Dynamic Execution
The right Stuff

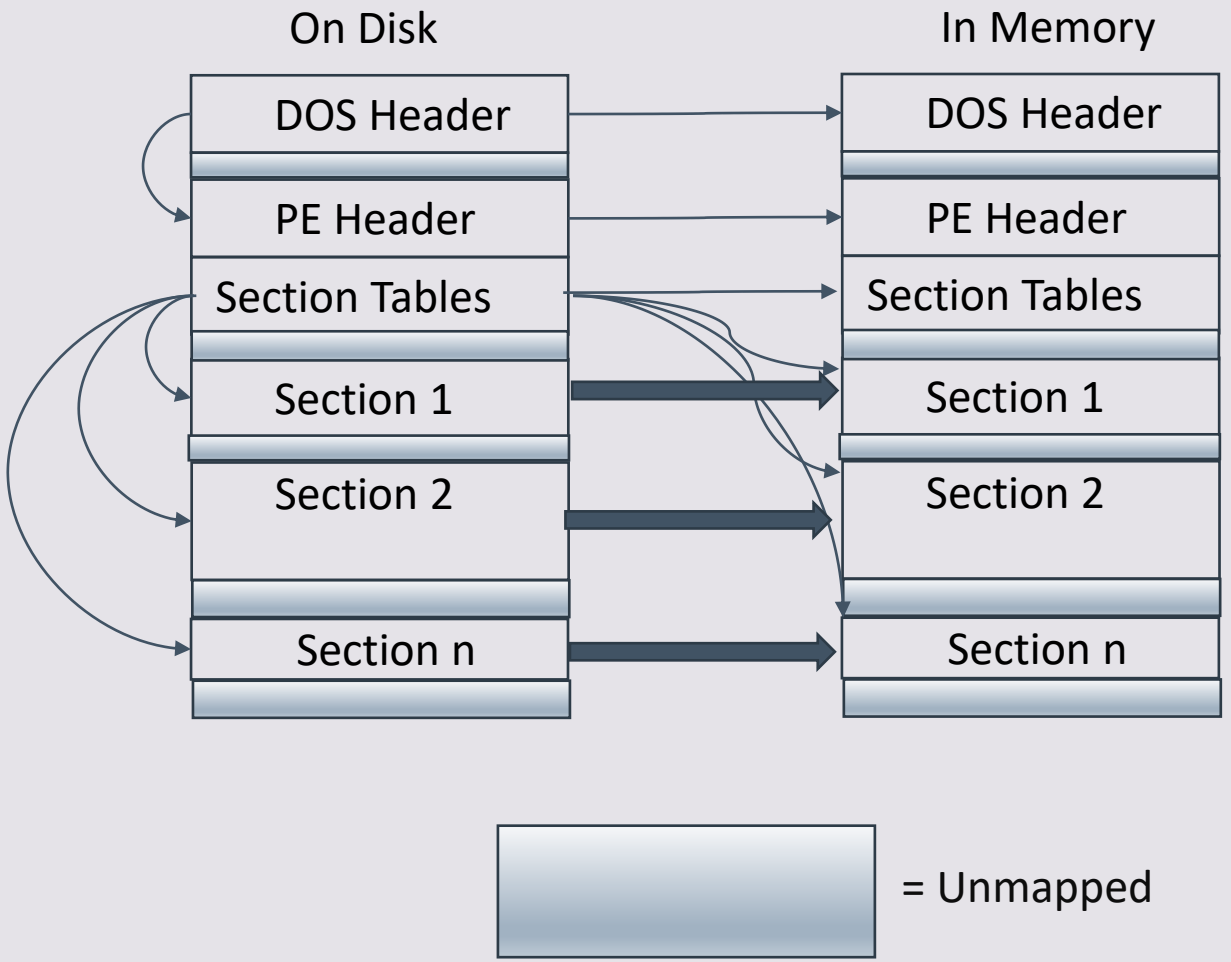
- Command Line Arguments
- Challenge/Response to external systems
  - Caution with concolic execution if a hash
- Network paths
- Missing or incorrect DLL’s

Perils of Dynamic Execution
The right Stuff

- Analytic environment detection
  - Is only 192.168.0.0/16 available?
  - DNS expectations
  - Keyboard type
  - R U a VM?
- Challenge/Response
- Port Knocking
- Unsleeping

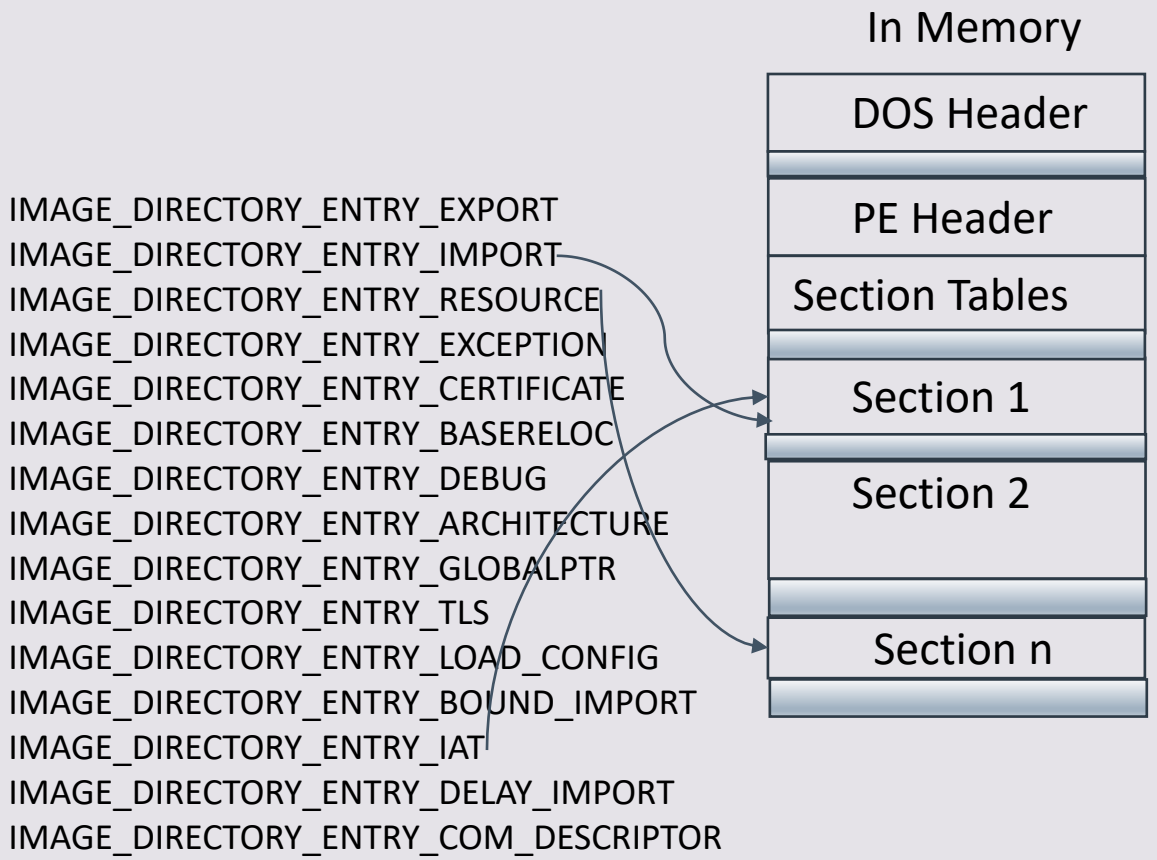
Perils of Dynamic Execution
Incomplete Reconstruction

- Brief PE Format tangent
  - File can extend beyond “PE”
  - Headers are loaded into Memory
  - Loader maps from disk into memory
  - Section Table contains
    - Offset, length On Disk
    - Offset (RVA), length In Memory
  - Sections
    - Not contiguous
    - Start on 2K (usually) boundary
    - Loader accepts overlaps, but who wins?
  - Sections do not have to map in order
  - Padding/End of File not mapped
  - Names are meaningless to loader



Perils of Dynamic Execution Incomplete Reconstruction

- Section Tables impact *LOADING* into memory (Where bytes go)
- Data Directories(in PE Optional Header) defines *USAGE* (What bytes mean)
  - Address (RVA)/ size of a table or string.
  - Imports, exports, resources, etc.
- No required correlation between sections and usage
- Can be NULL



Perils of Dynamic Execution Incomplete Reconstruction

- Sections are somewhat arbitrary
  - Yet another PE Header structure *Data Directories* contains offset, length for kind (i.e. import table)
  - Names can be almost anything < 8 characters
- Mind the Gap(s)/unmapped
  - Some malware uses unmapped (either interstitial or terminal) for config
  - Often ‘beyond’ PE file, but seen between DOS and PE Headers
- Punch Line – Once written to memory, original may not be reversable
  - Don’t know what Don’t know
  - Unmapped never written in memory, can’t reconstruct from memory image
- When unpacking creates new memory regions, how to ‘sectionize’

Getting ‘off-cut’

- Intel instructions are 1 to many bytes long
- A anti-disassembly technique is to jump into middle of multi-byte instruction
- Linear disassembly vs flow disassembly:

```
jmp short near ptr loc_2+1
; ---------------------------------------------
 loc_2:; CODE XREF: seg000:00000000j
              call near ptr 15FF2A71h
              or [ecx], dl
              inc eax
; ---------------------------------------------
              db 0
```

```
jmp short loc_3
; ---------------------------------
              db 0E8h
; ---------------------------------
 loc_3: ; CODE XREF: seg000:00000000j
              push 2Ah
              call Sleep
```

Some challenges

- ID dev shops
  - Practices
  - Tells, such as directory structures
- Handprints
  - ‘groups’ of functions that appear together
  - Shopping cart analogy
- Network Protocol
  - Reverse meanings
    - “Opcodes“ – choices, gaps, organization
    - Some way to document analysis and compare
  - Match client and servers
- Polyglot (Work in progress)

Some challenges

- Catch 22
  - Do we “see” X, when we don’t know how to look for it?
- Control Flow Graphs
  - Structural differences between ‘normal’ and ‘malicious’?
  - What features can be extracted for ML?
- Using disassembler (GHIDRA) ‘decisions’ as parameters
- Selection of features for different purposes:
  - Family
  - Function
  - Good/Bad
- Signature/Interpret blobs of code
  - Searching
  - Understanding

Some challenges

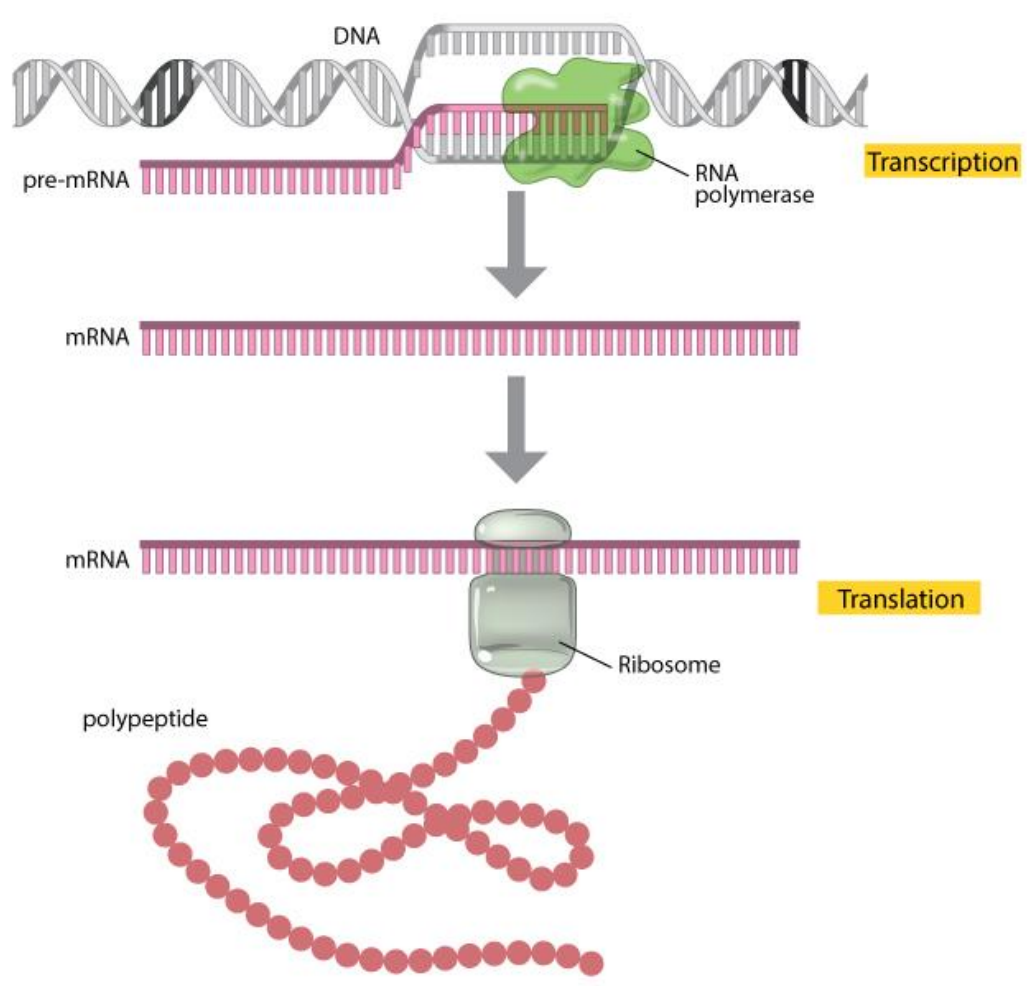
- Shellcode
  - Detection
  - Emulation
    - Usually needs some sort of environment
- How to prepare RE’s for changing environment
  - New processor (i.e. switch from x86 to x64, to ???)
  - Up/coming language (RUST, GO)

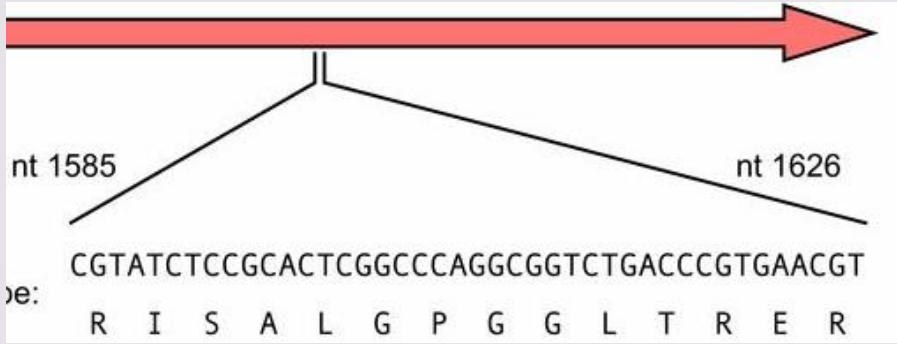
Questions / Discussion

POLYGLOT

- Where does “format Start?
  - Byte 0
  - Last byte
  - Within first 1024 bytes
- Find “hollow” space
- Multiple “zip”
  - MS Office
  - Jar
  - Android

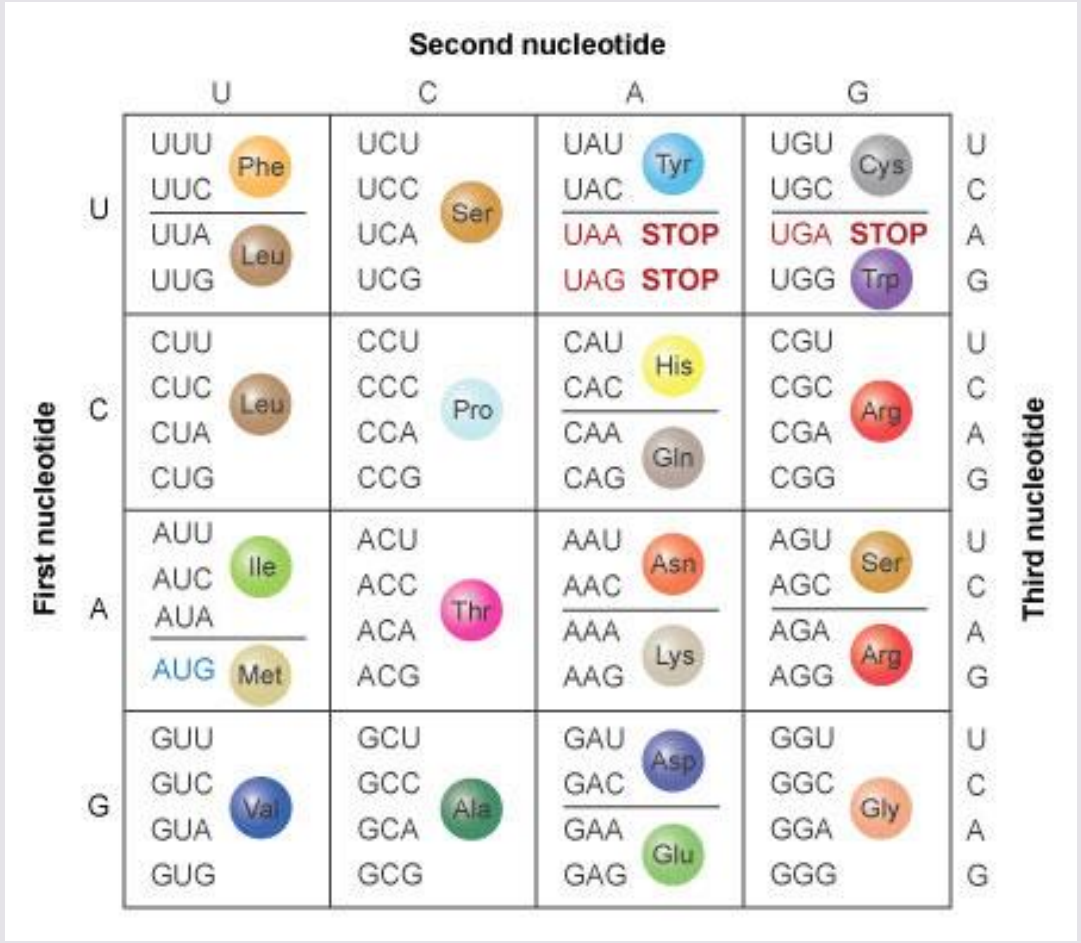
Microbiology Detour





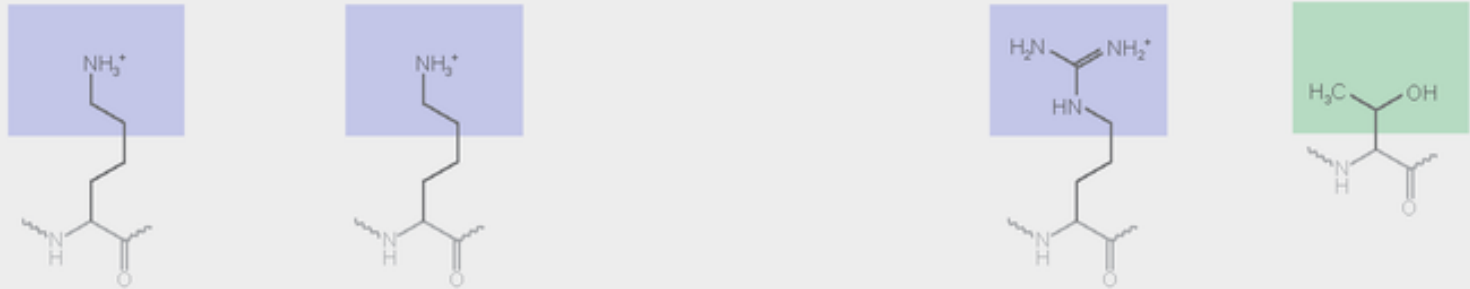
Microbiology Detour

- mRNA
  - Single strand of genes (A,C,U,G)
  - Three gene sequence (codons) translate to one of 20 amino acids
    - Fixed length instructions
  - Create a Protein (chain of peptides) that ‘run’ life’s processes
  - There’s a header and footer sequence

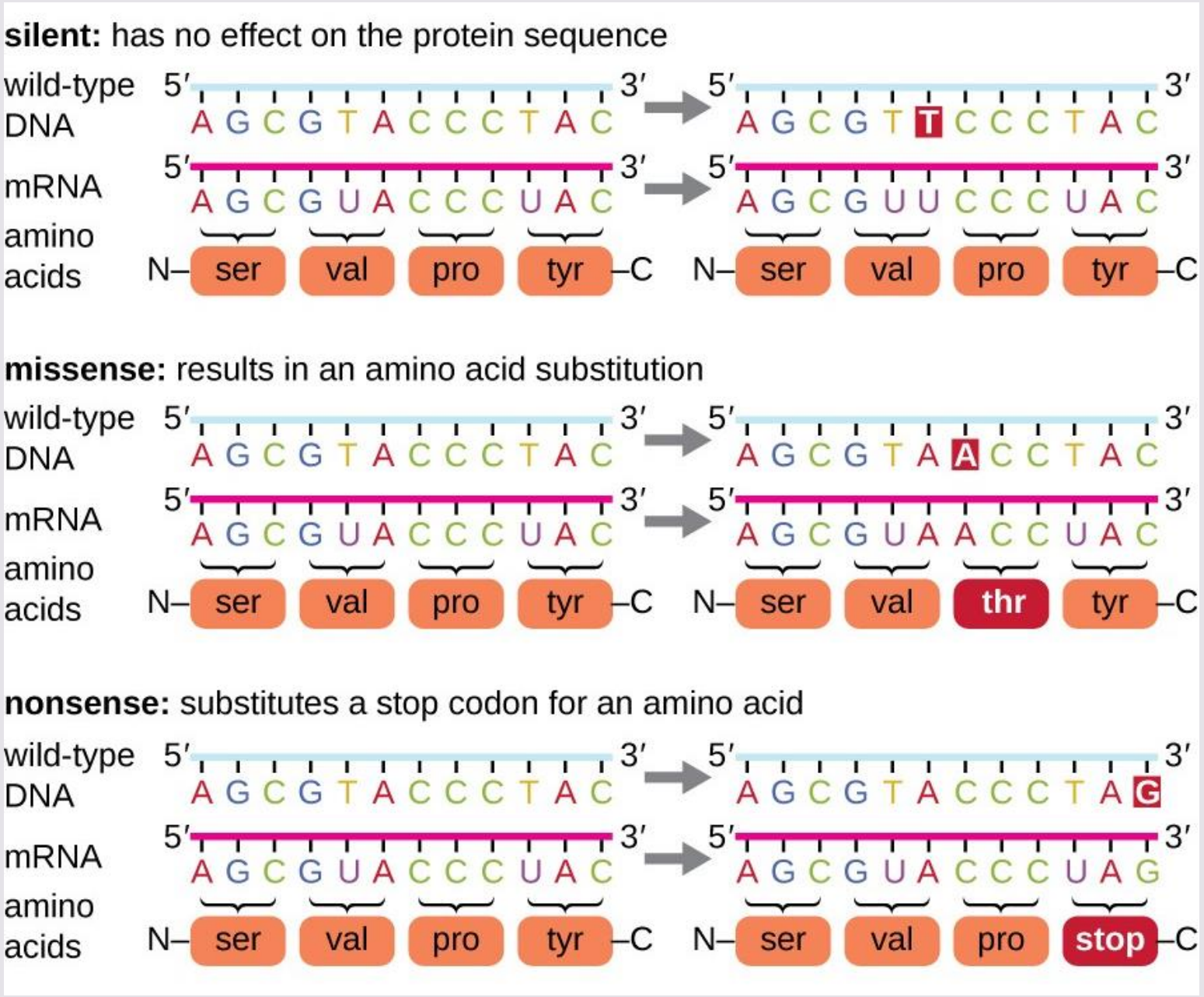


Mutation

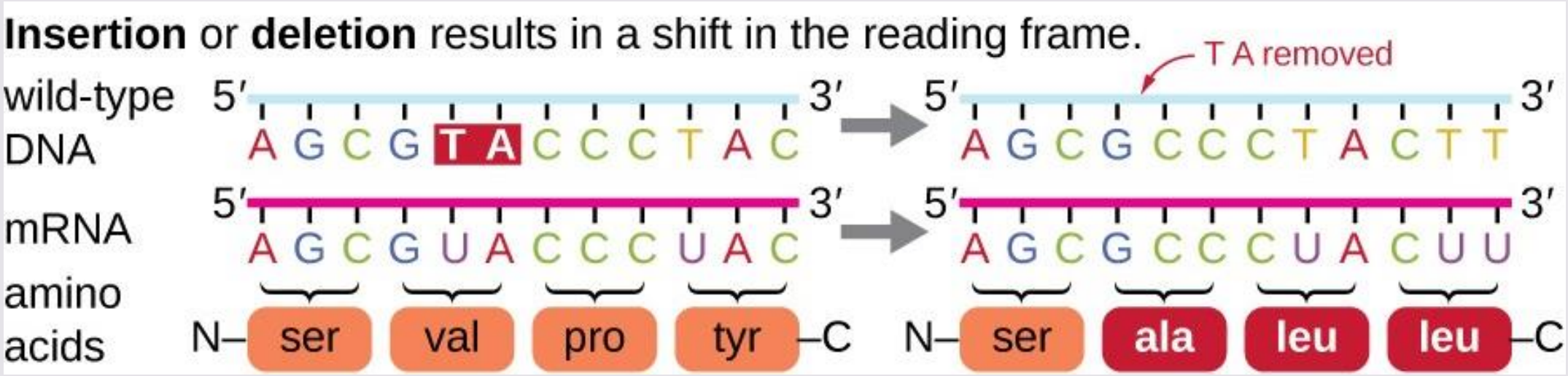
	No mutation	Point mutations			
		Silent	Nonsense	Missense	
				conservative	non-conservative
DNA level	TTC	TTT	ATC	TCC	TGC
mRNA level	AAG	AAA	UAG	AGG	ACG
protein level	**Lys**	**Lys**	**STOP**	**Arg**	**Thr**



Point Mutation



Frameshift


Insertion or deletion results in a shift in the reading frame.
T A removed
wild-type DNA
5′
3′
A G C G T A C C C T A C
A G C G C C C T A C T T
mRNA
A G C G U A C C C U A C
A G C G C C C U A C U U
amino acids
N–
ser
val
pro
tyr
–C
ala
leu


NATIONAL SECURITY AGENCY
UNITED STATES OF AMERICA


CENTRAL SECURITY SERVICE
UNITED STATES OF AMERICA

Thank you!