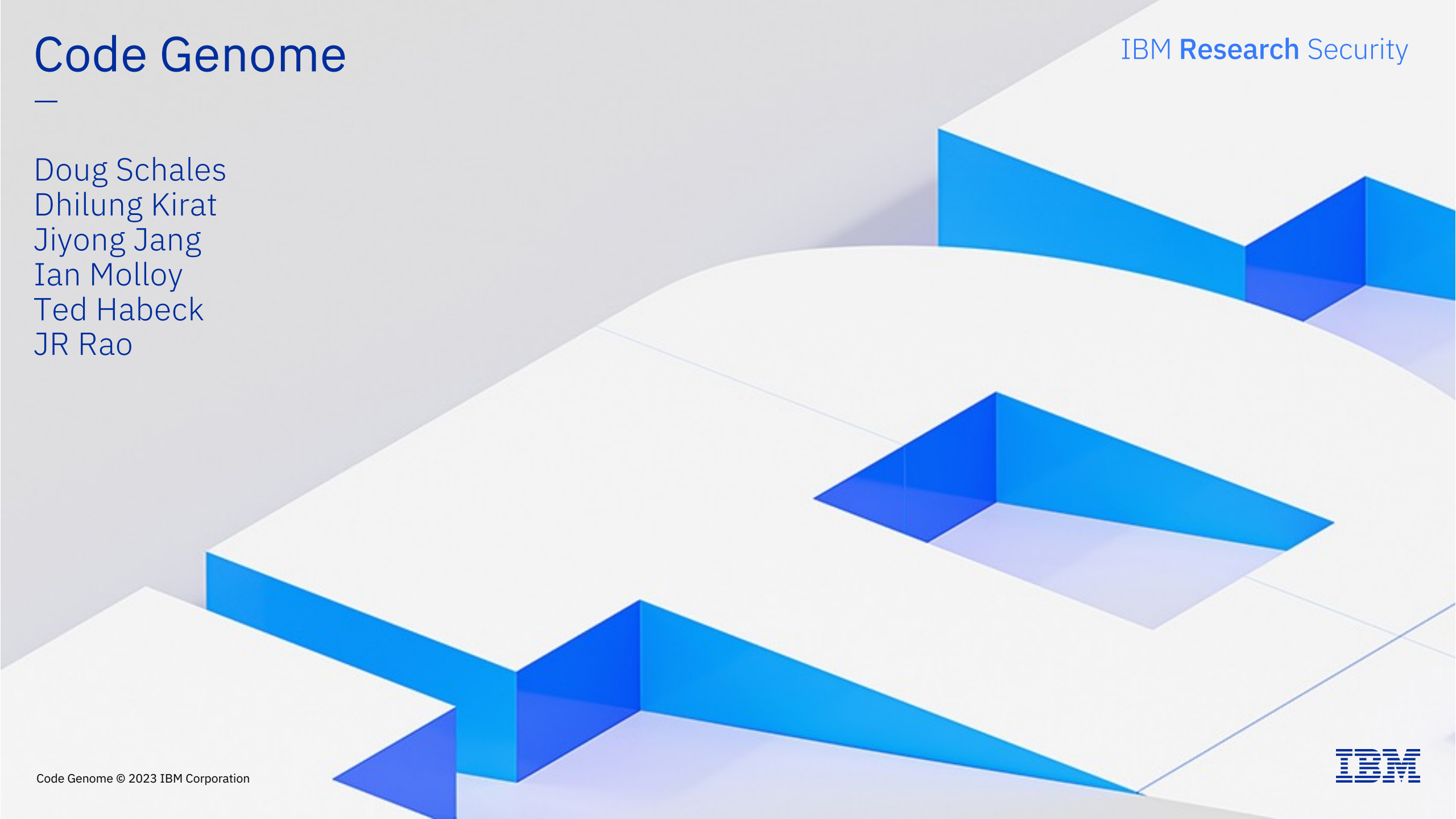
Code Genome

IBM **Research** Security

Doug Schales
Dhilung Kirat
Jiyong Jang
Ian Molloy
Ted Habeck
JR Rao



Code Genome = Code Gene + Knowledge Graph

```
; section: __text
; function: _f1 at 0x100000c20 -- 0x100000c35
0x100000c20:   55          push rbp
0x100000c21:   48 89 e5    mov rbp, rsp
0x100000c24:   89 7d fc    mov dword ptr [rbp - 4], edi
0x100000c27:   8b 7d fc    mov edi, dword ptr [rbp - 4]
0x100000c2a:   83 c7 20    add edi, 0x20
0x100000c2d:   89 7d f8    mov dword ptr [rbp - 8], edi
0x100000c30:   8b 45 f8    mov eax, dword ptr [rbp - 8]
0x100000c33:   5d          pop rbp
0x100000c34:   c3          ret
```
machine-code

```
int f1(int a){
  int x;
  x = a + 32;
  return x;
}
```
source code

Compile

Lift

```
define i64 @_f1(i32 %arg1) local_unnamed_addr {
dec_label_pc_100000c20:
  %tmp5 = zext i32 %arg1 to i64
  %v2_100000c2a = add nuw nsw i64 %tmp5, 32
  %v17_100000c2a = and i64 %v2_100000c2a, 4294967295
  ret i64 %v17_100000c2a
}
```
"raw" IR

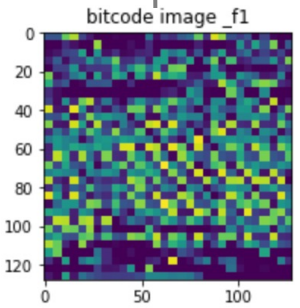
Canonicalize

```
define i64 @_F(i32 %a1) local_unnamed_addr #0 {
b1:
  %v0 = add i32 %a1, 32
  %v1 = zext i32 %v0 to i64
  ret i64 %v1
}
```
canonical IR

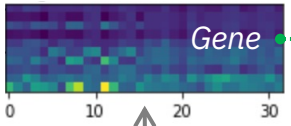
Convert

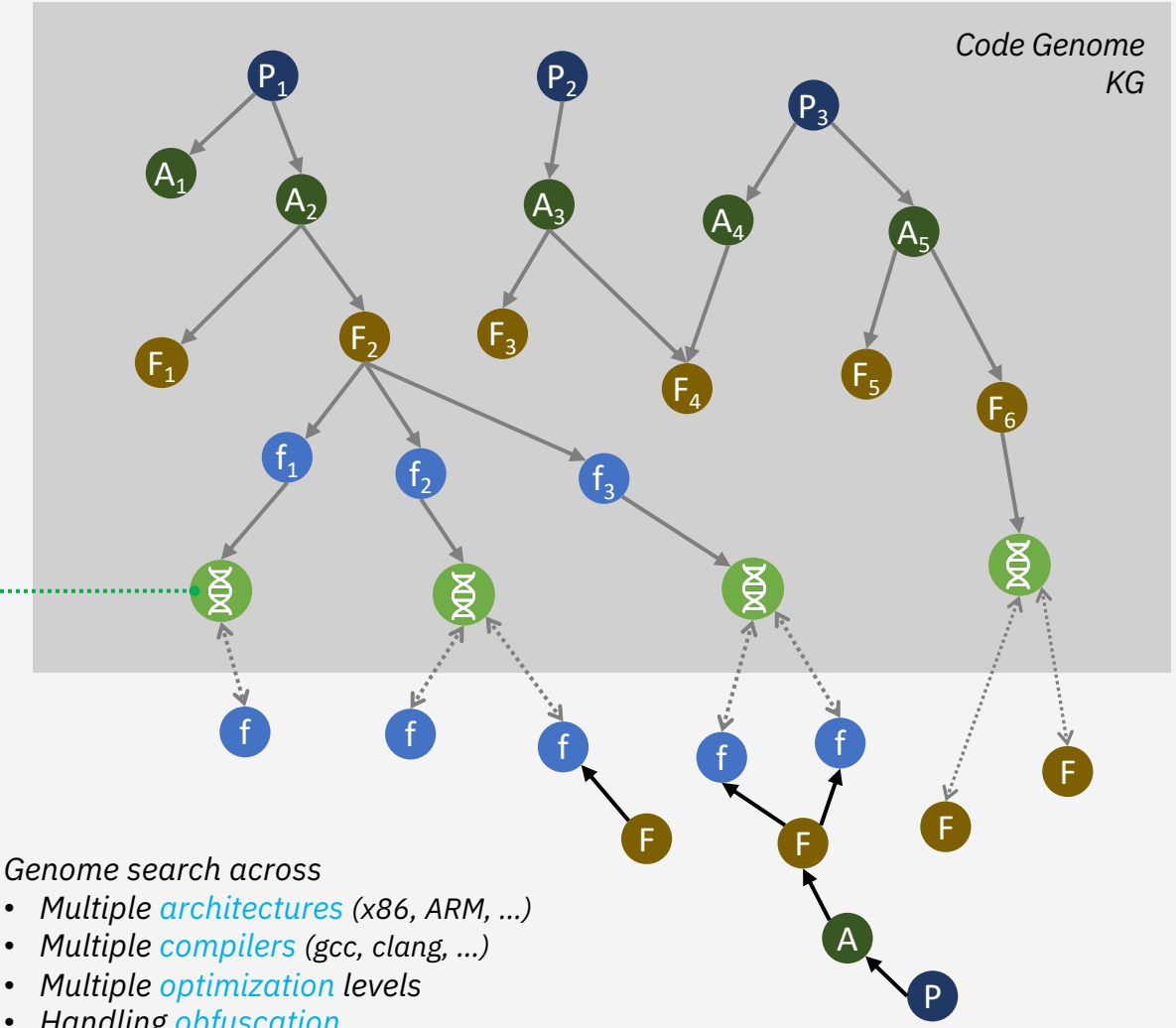
```
  Offset: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000: 42 43 C0 DE 35 14 00 00 05 00 00 00 62 0C 30 24   BC@^5.......b.0$
00000010: 49 59 BE 26 EF D3 3E 2D 44 01 32 05 00 00 00 00   IY>&oS>-D.2.....
00000020: 21 0C 00 00 D5 00 00 00 0B 02 21 00 02 00 00 00   !...U.....!.....
00000030: 16 00 00 00 07 81 23 91 41 C8 04 49 06 10 32 39   ......#.AH.I..29
00000040: 92 01 84 0C 25 05 08 19 1E 04 8B 62 80 0C 45 02   ....%......b..E.
00000050: 42 92 0B 42 64 10 32 14 38 08 18 4B 0A 32 32 88   B..Bd.2.8..K.22.
00000060: 48 70 C4 21 23 44 12 87 8C 10 41 92 02 64 C8 08   HpD!#D....A..dH.
00000070: B1 14 20 43 46 88 20 C9 01 32 32 84 18 2A 28 2A   1..CF..I.22..*(*
00000080: 90 31 7C B0 5C 91 20 C3 C8 00 00 00 89 20 00 00   .1|0\..CH.......
00000090: 0C 00 00 00 32 22 C8 08 20 64 85 04 93 21 A4 84   ....2"H..d...!$.
000000a0: 04 93 21 E3 84 A1 90 14 12 4C 86 8C 0B 84 64 4C   ..!......L....dL
000000b0: 10 18 73 04 A0 30 47 00 06 45 40 48 03 01 23 00   ..s..0G..E@H..#.
```
bitcode

Convert



Embedding



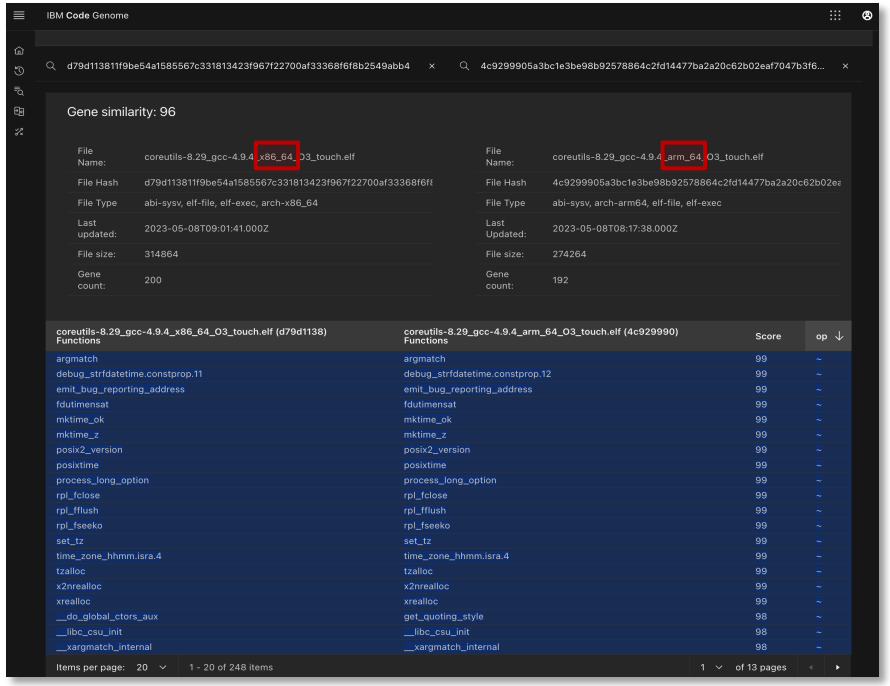


Genome search across

- *Multiple architectures (x86, ARM, ...)*
- *Multiple compilers (gcc, clang, ...)*
- *Multiple optimization levels*
- *Handling obfuscation*

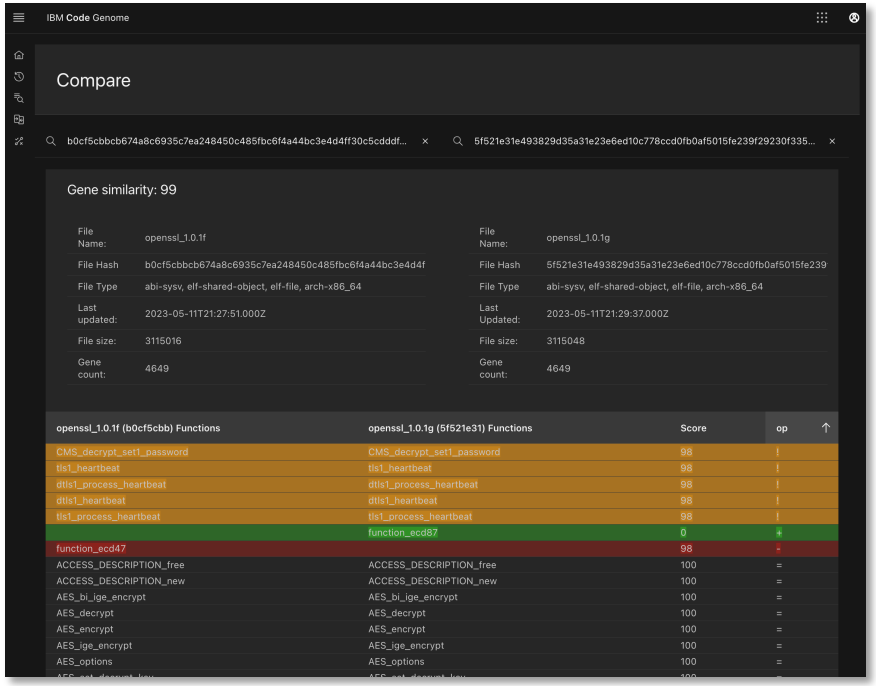
[1] Code Genome - Fingerprinting Code to Build Trustworthy SBOMs, Open Source Summit NA, 2023
[2] Code Genome: Fingerprinting Code to Help Secure the Software Supply Chain, Linux Foundation Member Summit, 2022

Code Genome Use Cases



Semantic search

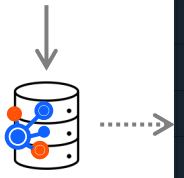
Same source code compiled in different environments and setups generates the same gene.



Code evolution

Examine the changes between versions and analyze how code has evolved over time.

Unknown package



Genome KG

Component	Version	License
arping	20160308	BSD and GPLv2+
clockdiff	20160308	BSD and GPLv2+
ifenslave	20160308	BSD and GPLv2+
iputils	20160308	BSD and GPLv2+
ping	20160308	BSD and GPLv2+
rdisc	20160308	BSD and GPLv2+
tracepath	20160308	BSD and GPLv2+
tracepath6	20160308	BSD and GPLv2+

Attribution & Forensics

Genome KG built on semantic fingerprinting allows identifying unknown binary code and understanding provenance and evolution.

- Cloud-native process engine and knowledge graph
- Currently supported
  - Binaries: `ELF, PE, Mach-O`
  - Architectures: `x86, x86_64, arm, aarch64, mips, ppc`
  - Packages: `deb, rpm, ipa`
  - Archives: `ar, cpio, tar, bzip2, gzip, zstd, xz, rar, 7zip`
- Demo is available at https://youtu.be/1JtaPY9TRfA