
Bitburner Documentation

Release 1.6.4

Bitburner

Jul 12, 2022

1	What is Bitburner?	3
1.1	Netscript	3
1.1.1	Learn to Program in Netscript	3
1.1.1.1	For Beginner Programmers	3
1.1.1.2	For Experienced Programmers	4
1.1.1.3	Netscript 1.0 vs Netscript 2.0	4
1.1.2	Netscript 1.0	4
1.1.2.1	Which ES6+ features are supported?	5
1.1.3	NS2	5
1.1.3.1	Browser compatibility	5
1.1.3.2	How to use ns2	5
1.1.3.3	Example	7
1.1.3.4	What's with the weird comment	7
1.1.4	Netscript Script Arguments	8
1.1.5	Netscript Basic Functions	9
1.1.5.1	hack() Netscript Function	9
1.1.5.2	grow() Netscript Function	10
1.1.5.3	weaken() Netscript Function	10
1.1.5.4	sleep() Netscript Function	11
1.1.5.5	print() Netscript Function	11
1.1.5.6	tprint() Netscript Function	11
1.1.5.7	disableLog() Netscript Function	12
1.1.5.8	enableLog() Netscript Function	12
1.1.5.9	isLogEnabled() Netscript Function	12
1.1.5.10	scan() Netscript Function	12
1.1.5.11	nuke() Netscript Function	13
1.1.5.12	brutessh() Netscript Function	13
1.1.5.13	ftpcrack() Netscript Function	13
1.1.5.14	relaysmtp() Netscript Function	13
1.1.5.15	httpworm() Netscript Function	14
1.1.5.16	sqlinject() Netscript Function	14
1.1.5.17	run() Netscript Function	14
1.1.5.18	exec() Netscript Function	15
1.1.5.19	spawn() Netscript Function	16
1.1.5.20	kill() Netscript Function	16
1.1.5.21	killall() Netscript Function	17

1.1.5.22	scp() Netscript Function	17
1.1.5.23	ls() Netscript Function	18
1.1.5.24	ps() Netscript Function	18
1.1.5.25	hasRootAccess() Netscript Function	19
1.1.5.26	getHackingLevel() Netscript Function	19
1.1.5.27	getHackingMultipliers() Netscript Function	19
1.1.5.28	getHacknetMultipliers() Netscript Function	20
1.1.5.29	getServerMoneyAvailable() Netscript Function	20
1.1.5.30	getServerMaxMoney() Netscript Function	21
1.1.5.31	getServerSecurityLevel() Netscript Function	21
1.1.5.32	getServerMinSecurityLevel() Netscript Function	21
1.1.5.33	getServerRequiredHackingLevel() Netscript Function	21
1.1.5.34	getServerNumPortsRequired() Netscript Function	22
1.1.5.35	getServerMaxRam() Netscript Function	22
1.1.5.36	getServerUsedRam() Netscript Function	22
1.1.5.37	serverExists() Netscript Function	22
1.1.5.38	fileExists() Netscript Function	23
1.1.5.39	isRunning() Netscript Function	23
1.1.5.40	getPurchasedServerCost() Netscript Function	24
1.1.5.41	purchaseServer() Netscript Function	24
1.1.5.42	deleteServer() Netscript Function	25
1.1.5.43	getPurchasedServers() Netscript Function	25
1.1.5.44	getPurchasedServerLimit() Netscript Function	25
1.1.5.45	getPurchasedServerMaxRam() Netscript Function	26
1.1.5.46	scriptRunning() Netscript Function	26
1.1.5.47	scriptKill() Netscript Function	26
1.1.5.48	getScriptRam() Netscript Function	27
1.1.6	Netscript Advanced Functions	27
1.1.6.1	autocomplete() Netscript Function	27
1.1.6.2	Injecting HTML in the game	28
1.1.7	Netscript Hacknet Node API	28
1.1.7.1	numNodes() Netscript Function	29
1.1.7.2	purchaseNode() Netscript Function	29
1.1.7.3	getPurchaseNodeCost() Netscript Function	29
1.1.7.4	getNodeStats() Netscript Function	29
1.1.7.5	upgradeLevel() Netscript Function	30
1.1.7.6	upgradeRam() Netscript Function	30
1.1.7.7	upgradeCore() Netscript Function	30
1.1.7.8	getLevelUpgradeCost() Netscript Function	31
1.1.7.9	getRamUpgradeCost() Netscript Function	31
1.1.7.10	getCoreUpgradeCost() Netscript Function	31
1.1.7.11	Referencing a Hacknet Node	32
1.1.7.12	RAM Cost	32
1.1.7.13	Utilities	32
1.1.7.14	Example(s)	32
1.1.8	Netscript Miscellaneous	33
1.1.8.1	Netscript Ports	33
1.1.8.2	Comments	35
1.1.8.3	Importing Functions	35
1.1.8.4	Standard, Built-In JavaScript Objects	36
1.2	Basic Gameplay	36
1.2.1	Stats	36
1.2.1.1	Hacking	36
1.2.1.2	Strength	37

1.2.1.3	Defense	37
1.2.1.4	Dexterity	38
1.2.1.5	Agility	38
1.2.1.6	Charisma	38
1.2.2	Terminal	39
1.2.2.1	Filesystem (Directories)	39
1.2.2.1.1	Directories	39
1.2.2.1.2	Absolute vs Relative Paths	39
1.2.2.1.3	Netscript	40
1.2.2.1.4	Missing Features	40
1.2.2.2	Commands	40
1.2.2.2.1	alias	40
1.2.2.2.2	analyze	41
1.2.2.2.3	backdoor	41
1.2.2.2.4	buy	41
1.2.2.2.5	cat	41
1.2.2.2.6	cd	42
1.2.2.2.7	check	42
1.2.2.2.8	clear/cls	42
1.2.2.2.9	connect	42
1.2.2.2.10	download	42
1.2.2.2.11	expr	43
1.2.2.2.12	free	43
1.2.2.2.13	hack	43
1.2.2.2.14	help	43
1.2.2.2.15	home	43
1.2.2.2.16	hostname	43
1.2.2.2.17	ifconfig	44
1.2.2.2.18	kill	44
1.2.2.2.19	killall	44
1.2.2.2.20	ls	44
1.2.2.2.21	lscpu	45
1.2.2.2.22	mem	45
1.2.2.2.23	mv	45
1.2.2.2.24	nano	45
1.2.2.2.25	ps	45
1.2.2.2.26	rm	46
1.2.2.2.27	run	46
1.2.2.2.28	scan	46
1.2.2.2.29	scan-analyze	46
1.2.2.2.30	scp	47
1.2.2.2.31	sudov	47
1.2.2.2.32	tail	47
1.2.2.2.33	top	47
1.2.2.2.34	unalias	47
1.2.2.2.35	wget	47
1.2.2.3	Argument Parsing	48
1.2.2.4	Chaining Commands	48
1.2.3	Servers	49
1.2.3.1	Server RAM	49
1.2.3.2	Identifying Servers	49
1.2.3.3	Player-owned Servers	49
1.2.3.4	Hackable Servers	49
1.2.4	Hacking	49

1.2.4.1	Gaining Root Access	50
1.2.4.2	General Hacking Mechanics	50
1.2.4.3	Server Security	51
1.2.4.4	Backdoors	51
1.2.5	Scripts	51
1.2.5.1	Script Arguments	51
1.2.5.2	Identifying a Script	51
1.2.5.3	Multithreading scripts	52
1.2.5.4	Working with Scripts in Terminal	52
1.2.5.5	Working with Scripts in Netscript	53
1.2.5.6	Notes about how Scripts Work Offline	53
1.2.6	World	54
1.2.7	Factions	54
1.2.7.1	List of Factions and their Requirements	55
1.2.8	Augmentations	59
1.2.8.1	How to acquire Augmentations	59
1.2.8.2	Installing Augmentations	59
1.2.8.3	Purchasing Multiple Augmentations	60
1.2.9	Companies	60
1.2.9.1	Information about all Companies	60
1.2.10	Crimes	61
1.2.10.1	Basic Mechanics	61
1.2.10.2	Crime details	61
1.2.11	Infiltration	61
1.2.11.1	Overview	61
1.2.12	Stock Market	62
1.2.12.1	Fundamentals	62
1.2.12.1.1	Positions: Long vs Short	62
1.2.12.1.2	Forecast & Second-Order Forecast	63
1.2.12.1.3	Spread (Bid Price & Ask Price)	63
1.2.12.1.4	Transactions Influencing Stock Forecast	63
1.2.12.1.5	Order Types	63
1.2.12.1.6	Player Actions Influencing Stocks	64
1.2.12.2	Automating the Stock Market	65
1.2.12.3	Under the Hood	65
1.2.12.4	Offline Progression	65
1.2.13	Coding Contracts	65
1.2.13.1	Running in Terminal	65
1.2.13.2	Interacting through Scripts	66
1.2.13.3	Submitting Solutions	66
1.2.13.4	Rewards	66
1.2.13.5	Notes	66
1.2.13.6	List of all Problem Types	66
1.3	Advanced Gameplay	68
1.3.1	BitNodes	68
1.3.1.1	What is a BitNode	68
1.3.1.2	How to destroy a BitNode	68
1.3.2	Source-Files	69
1.3.2.1	List of all Source-Files	71
1.3.3	Intelligence	72
1.3.4	Sleeves	72
1.3.4.1	Duplicate Sleeves	72
1.3.4.1.1	Obtaining Duplicate Sleeves	72
1.3.4.1.2	Synchronization	73

	1.3.4.1.3	Sleeve Shock	73
	1.3.4.1.4	Augmentations	73
	1.3.4.1.5	Memory	73
1.3.5	Grafting		73
1.3.6	Hacking algorithms		74
	1.3.6.1	Self-contained algorithms	74
	1.3.6.2	Loop algorithms	75
	1.3.6.3	Batch algorithms (HGW, HWGW, or Cycles)	75
1.4	Keyboard Shortcuts		77
	1.4.1	Game Navigation	77
	1.4.2	Terminal Shortcuts	78
	1.4.3	Terminal Bash Shortcuts	78
	1.4.4	Popup/Dialog Box Shortcuts	79
1.5	Game Frozen or Stuck?		79
	1.5.1	Infinite Loop in NetscriptJS	79
	1.5.2	Bug	79
1.6	Guides & Tips		79
	1.6.1	Getting Started Guide for Beginner Programmers	80
	1.6.1.1	Introduction	80
	1.6.1.2	First Steps	80
	1.6.1.3	Creating our First Script	80
	1.6.1.4	Running our Scripts	82
	1.6.1.5	Increasing Hacking Level	85
	1.6.1.6	Editing our Hacking Script	85
	1.6.1.7	Creating a New Script to Purchase New Servers	86
	1.6.1.8	Additional Sources of Income	87
	1.6.1.8.1	Hacknet Nodes	87
	1.6.1.8.2	Crime	87
	1.6.1.8.3	Work for a Company	88
	1.6.1.9	After you Purchase your New Servers	88
	1.6.1.10	Reaching a Hacking Level of 50	88
	1.6.1.10.1	Creating your first program: BruteSSH.exe	88
	1.6.1.10.2	Optional: Create AutoLink.exe	89
	1.6.1.10.3	Joining your first faction: CyberSec	89
	1.6.1.11	Using Additional Servers to Hack Joesguns	90
	1.6.1.11.1	Copying our Scripts	90
	1.6.1.12	Profiting from Scripts & Gaining Reputation with CyberSec	91
	1.6.1.13	Purchasing Upgrades and Augmentations	91
	1.6.1.13.1	Upgrading RAM on Home computer	91
	1.6.1.13.2	Purchasing your First Augmentations	91
	1.6.1.14	Next Steps	92
	1.6.1.14.1	Installing Augmentations (and Resetting)	92
	1.6.1.14.2	Automating the Script Startup Process	92
	1.6.1.15	Random Tips	93
1.6.2	What BitNode should I do?		94
	1.6.2.1	Overview of each BitNode	94
	1.6.2.1.1	BitNode-1: Source Genesis	94
	1.6.2.1.2	BitNode-2: Rise of the Underworld	94
	1.6.2.1.3	BitNode-3: Corporatocracy	95
	1.6.2.1.4	BitNode-4: The Singularity	95
	1.6.2.1.5	BitNode-5: Artificial Intelligence	96
	1.6.2.1.6	BitNode-6: Bladeburners	96
	1.6.2.1.7	BitNode-7: Bladeburners 2079	97
	1.6.2.1.8	BitNode-8: Ghost of Wall Street	98

	1.6.2.1.9	BitNode-9: Hacktocracy	98
	1.6.2.1.10	BitNode-10: Digital Carbon	99
	1.6.2.1.11	BitNode-11: The Big Crash	99
	1.6.2.1.12	BitNode-12: The Recursion	100
	1.6.2.2	Recommended BitNodes	100
	1.6.2.2.1	For fast progression	100
	1.6.2.2.2	For the strongest Source-Files	101
	1.6.2.2.3	For more scripting/hacking	101
	1.6.2.2.4	For new mechanics	102
	1.6.2.2.5	For a Challenge	102
1.7		Tools & Resources	102
	1.7.1	Official Script Repository	102
	1.7.2	Visual Studio Code Extension	102
1.8		Changelog	103
	1.8.1	v1.6.3 - 2022-04-01 Few stanek fixes	103
	1.8.2	v1.6.0 - 2022-03-29 Grafting	103
	1.8.3	v1.5.0 - Steam Cloud integration	105
	1.8.4	v1.4.0 - 2022-01-18 Sharing is caring	107
	1.8.5	v1.3.0 - 2022-01-04 Cleaning up	110
	1.8.6	v1.1.0 - 2021-12-18 You guys are awesome (community because they're god damn awesome)	112
	1.8.7	v1.1.0 - 2021-12-03 BN13: They're Lunatics (hydroflame & community)	114
	1.8.8	v1.0.2 - 2021-11-17 It's the little things (hydroflame)	115
	1.8.9	v1.0.1 - 2021-11-17 New documentation (hydroflame)	116
	1.8.10	v1.0.0 - 2021-11-10 Breaking the API :((blame hydroflame)	117
	1.8.11	v0.58.0 - 2021-10-27 Road to Steam (hydroflame & community)	118
	1.8.12	v0.57.0 - 2021-10-16 It was too cheap! (hydroflame & community)	119
	1.8.13	v0.56.0 - 2021-10-11 Trimming the backlog (hydroflame & community)	120
	1.8.14	v0.55.0 - 2021-09-20 Material UI (hydroflame & community)	122
	1.8.15	v0.54.0 - 2021-09-20 One big react node (hydroflame & community)	123
	1.8.16	v0.53.0 - 2021-09-09 Way too many things. (hydroflame & community)	123
	1.8.17	v0.52.9 - 2021-08-27 Less lag! (hydroflame & community)	125
	1.8.18	v0.52.8 - 2021-08-23 Fixing the previous patch tbh ROUND 2 (hydroflame)	125
	1.8.19	v0.52.7 - 2021-08-21 Fixing the previous patch tbh (hydroflame)	126
	1.8.20	v0.52.6 - 2021-08-21 Logboxes and VS-code (hydroflame)	126
	1.8.21	v0.52.5 - 2021-08-19 CPU cores are useful!? (hydroflame)	127
	1.8.22	v0.52.4 - 2021-08-19 Bladeburner in React (hydroflame)	127
	1.8.23	v0.52.3 - 2021-08-15 Gangs were OP (hydroflame)	128
	1.8.24	v0.52.2 - 2021-08-15 Oh yeah, BN11 is a thing (drunk hydroflame tbh)	128
	1.8.25	v0.52.1 - 2021-08-10 bugfixing (hydroflame & community)	128
	1.8.26	v0.52.0 - 2021-06-13 Infiltration 2.0 (hydroflame & community)	129
	1.8.27	v0.51.10 - 2021-05-31 Focus Mark, Focus! (hydroflame)	130
	1.8.28	v0.51.9 - 2021-05-17 offline progress and exports! (hydroflame & community)	130
	1.8.29	v0.51.8 - 2021-05-07 It was there all along (hydroflame & community)	131
	1.8.30	v0.51.7 - 2021-04-28 n00dles (hydroflame & community)	132
	1.8.31	v0.51.6 - 2021-04-28 Backdoor! (hydroflame & community)	134
	1.8.32	v0.51.5 - 2021-04-20 Flags! (hydroflame)	134
	1.8.33	v0.51.4 - 2021-04-19 Manual hacking is fun (hydroflame)	135
	1.8.34	v0.51.3 - 2021-04-16 Y'all broke it on the first day (hydroflame)	136
	1.8.35	v0.51.2 - 2021-04-09 Vegas, Baby! (hydroflame)	136
	1.8.36	v0.51.1 - 2021-04-06 Bugfixes because the author of the last patch sucks (it's hydroflame)	137
	1.8.37	v0.51.0 - 2021-03-31 Formulas (hydroflame)	137
	1.8.38	v0.50.2 - 2021-03-25 Everyone asked for this one. (hydroflame)	138
	1.8.39	v0.50.1 - 2021-03-22 (hydroflame)	138
	1.8.40	v0.50.0 - 2021-03-20 Intelligence (hydroflame)	138

1.8.41	v0.49.2 - 2021-03-13 (hydroflame)	139
1.8.42	v0.49.0 - 2021-03-11 Source-File -1 (hydroflame)	139
1.8.43	v0.48.0 - ASCII - 2021-03-07 (hydroflame)	140
1.8.44	v0.47.2 - 7/15/2019	141
1.8.45	v0.47.1 - 6/27/2019	141
1.8.46	v0.47.0 - 5/17/2019	142
1.8.47	v0.46.3 - 4/20/2019	143
1.8.48	v0.46.2 - 4/14/2019	143
1.8.49	v0.46.1 - 4/12/2019	144
1.8.50	v0.46.0 - 4/3/2019	144
1.8.51	v0.45.1 - 3/23/2019	144
1.8.52	v0.45.0 - 3/22/2019	145
1.8.53	v0.44.1 - 3/4/2019	145
1.8.54	v0.44.0 - 2/26/2019	146
1.8.55	v0.43.1 - 2/11/2019	147
1.8.56	v0.43.0 - 2/4/2019	148
1.8.57	v0.42.0 - 1/8/2019	148
1.8.58	v0.41.2 - 11/23/2018	149
1.8.59	v0.41.1 - 11/5/2018	150
1.8.60	v0.41.0 - 10/29/2018	151
1.8.61	v0.40.5 - 10/09/2018	152
1.8.62	v0.40.4 - 9/29/2018	152
1.8.63	v0.40.3 - 9/15/2018	152
1.8.64	v0.40.2 - 8/27/2018	153
1.8.65	v0.40.1 - 8/5/2018 - Community Update	154
1.8.66	v0.40.0 - 7/28/2018	154
1.8.67	v0.39.1 - 7/4/2018	155
1.8.68	v0.39.0 - 6/25/2018	156
1.8.69	v0.38.1 - 6/15/2018	156
1.8.70	v0.38.0 - 6/12/2018	157
1.8.71	v0.37.2 - 6/2/2018	157
1.8.72	v0.37.1 - 5/22/2018	157
1.8.73	v0.37.0 - 5/20/2018	158
1.8.74	v0.36.1 - 5/11/2018	158
1.8.75	v0.36.0 - 5/2/2018	159
1.8.76	v0.35.2 - 3/26/2018	159
1.8.77	v0.35.1 - 3/12/2018	159
1.8.78	v0.35.0 - 3/3/2018	160
1.8.79	v0.34.5 - 2/24/2018	161
1.8.80	v0.34.4 - 2/14/2018	162
1.8.81	v0.34.3 - 1/31/2018	162
1.8.82	v0.34.2 - 1/27/2018	163
1.8.83	v0.34.1 - 1/19/2018	163
1.8.84	v0.34.0 - 12/6/2017	164
1.8.85	v0.33.0 - 12/1/2017	164
1.8.86	v0.32.1 - 11/2/2017	165
1.8.87	v0.32.0 - 10/25/2017	165
1.8.88	v0.31.0 - 10/15/2017	165
1.8.89	v0.30.0 - 10/9/2017	166
1.8.90	v0.29.3 - 10/3/2017	166
1.8.91	v0.29.2 - 10/1/2017	166
1.8.92	v0.29.1 - 9/27/2017	167
1.8.93	v0.29.0 - 9/19/2017	167
1.8.94	v0.28.6 - 9/15/2017	167

1.8.95	v0.28.5 - 9/13/2017	168
1.8.96	v0.28.4 - 9/11/2017	168
1.8.97	v0.28.3 - 9/7/2017	168
1.8.98	v0.28.2 - 9/4/2017	168
1.8.99	v0.28.1 - 9/1/2017	169
1.8.100	v0.28.0 - 8/30/2017	169
1.8.101	v0.27.3 - 8/19/2017	169
1.8.102	v0.27.2 - 8/18/2017	169
1.8.103	v0.27.1 - 8/15/2017	169
1.8.104	v0.27.0 - 8/13/2017	170
1.8.105	v0.26.4 - 8/1/2017	170
1.8.106	v0.26.3	170
1.8.107	v0.26.2	170
1.8.108	v0.26.1	171
1.8.109	v0.26.0	171
1.8.110	v0.25.0	172
1.8.111	v0.24.1	173
1.8.112	v0.24.0	173
1.8.113	v0.23.1	174
1.8.114	v0.23.0	174
1.8.115	v0.22.1	174
1.8.116	v0.22.0 - Major rebalancing, optimization, and favor system	174
1.8.117	v0.21.1	175
1.8.118	v0.21.0	175
1.8.119	v0.20.2	176
1.8.120	v0.20.1	176
1.8.121	v0.20.0	177
1.8.122	v0.19.7	177
1.8.123	v0.19.6	177
1.8.124	v0.19.0	178
1.8.125	v0.18.0	178
1.8.126	v0.17.1	179
1.8.127	v0.17.0	179
1.8.128	v0.16.0	179
1.8.129	v0.15.0	180
1.9	v1.0.0 Migration Guide	180
1.10	Hi there, hello	181

2 Indices and tables 183

Index 185

Bitburner is a programming-based [incremental game](#) that revolves around hacking and cyberpunk themes. The game is currently in the early beta stage of development. [It can be played here.](#)

CHAPTER 1

What is Bitburner?

Bitburner is a cyberpunk-themed incremental RPG where you, the player, take the role of an unknown hacker in a dark, dystopian world. When a mysterious hacker called jump3R messages you, he/she confirms your suspicions that there is something wrong with the world around you. Now, aided by jump3R, you embark on a quest to gain money and power by any means necessary, in the hopes that this will lead to uncover the secrets that you've been searching for.

1.1 Netscript

Netscript is the programming language used in the world of Bitburner.

When you write scripts in Bitburner, they are written in the Netscript language. Netscript is simply a subset of [JavaScript](#). This means that Netscript's syntax is identical to that of JavaScript, but it does not implement some of the features that JavaScript has.

1.1.1 Learn to Program in Netscript

Netscript is simply a subset of [JavaScript](#), with some additional functions added in to allow interaction with the game.

1.1.1.1 For Beginner Programmers

If you have little to no programming experience, that's okay! You don't need to be a great programmer in order to enjoy or play this game. In fact, this game could help you learn some basic programming concepts.

Here are some good tutorials for learning programming/JavaScript as a beginner:

- [Learn-JS](#)
- [programiz](#)
- [Speaking JavaScript](#) This is a bit on the longer side. You can skip all of the historical background stuff. Recommended chapters: 1, 7-18

1.1.1.2 For Experienced Programmers

The following section lists several good tutorials/resources for those who have experience programming but who have not worked extensively with JavaScript before.

Before that, however, it's important to clarify some terminology about the different versions of JavaScript. These are summarized in this article:

[WTF is ES6, ES8, ES2017, ECMAScript...](#)

An important takeaway from this article is that ES6, also known as ES2015, introduced many major features that are commonly seen in modern JavaScript programming. However, this means that ES5 engines and interpreters will fail if they encounter these ES6 features. You'll see why this is important further down.

- [MDN Introduction to JS](#)
- [Eloquent JavaScript \(ES6+\)](#) Recommended Chapters: Introduction, 1-6
- [Modern Javascript Tutorial \(ES6+\)](#) Recommended Chapters: 2, 4-6

1.1.1.3 Netscript 1.0 vs Netscript 2.0

There are two versions of Netscript:

- [Netscript 1.0](#)
- [NS2](#)

Visit the pages above to get more details about each version. If you are new to programming or unfamiliar with JavaScript, I would recommend starting out with [Netscript 1.0](#). Experienced web developers can use [NS2](#) to take advantage of faster speeds and additional features.

Here is a short summary of the differences between Netscript 1.0 and Netscript 2.0:

Netscript 1.0

- ES5
- Some ES6 features implemented with polyfills
- Slow compared to NetscriptJS (interpreter runs at the "Netscript Exec Time" speed configured in options)
- Compatible with all browsers

Netscript JS (Netscript 2.0)

- Supports (almost) all features of modern JavaScript
- Extremely fast - code is executed as an Async Function
- Works on most modern browsers.
- Each script becomes a module and therefore all instances of that script can easily share data between each other (essentially global/static variables)

1.1.2 Netscript 1.0

Netscript 1.0 is implemented using a modified version of Neil Fraser's [JS-Interpreter](#).

This is an ES5 JavaScript interpreter. This means that (almost) any JavaScript feature that is available in ES5 is also available in Netscript 1.0. However, this also means that the interpreter does not natively support any JavaScript features introduced in versions ES6 or after.

If you are confused by the ES5/ES6/etc. terminology, consider reading this: [WTF is ES6, ES8, ES2017, ECMAScript...](#)

Netscript 1.0 scripts end with the “.script” extension in their filenames.

1.1.2.1 Which ES6+ features are supported?

Netscript 1.0 is a ES5 interpreter, but the following features from versions ES6 and above are supported as well.

- import - See *Importing Functions*
- Array
- Array find()
- Array findIndex()
- Array includes()
- String
- String endsWith()
- String includes()
- String startsWith()

1.1.3 NS2

The improved version of Netscript that allows users to write full-fledged Javascript code in their scripts, while still being able to access the Netscript functions.

ns2 was developed primarily by [Github user jaguilar](#)

On top of having almost all of the features and capabilities of JavaScript, ns2 is also significantly faster than ns1.

This documentation will not go over any of the additional features of ns2, since there is plenty of documentation on Javascript available on the web.

1.1.3.1 Browser compatibility

As of the time of writing this, a few browsers do not support [dynamic import](#) functionality and therefore cannot run ns2 scripts. These browsers will thus only be capable of using ns1.

1.1.3.2 How to use ns2

Working with ns2 scripts is the same as ns1 scripts. The only difference is that ns2 scripts use the “.js” extension rather than “.script”. E.g.:

```
$ nano foo.js
$ run foo.js -t 100 arg1 arg2 arg3
exec("foo.js", "purchasedServer1", "100", "randomArg");
```

The caveat when using ns2 to write scripts is that your code must be asynchronous. Furthermore, instead of using the global scope and executing your code sequentially, ns2 uses a `main()` function as an entry point.

Furthermore, the “Netscript environment” must be passed into a ns2 script through the main function. This environment includes all of the pre-defined Netscript functions (`hack()`, `exec`, etc.) as well as the arguments you pass to the script.

Therefore, the signature of the `main()` function must be:

```
export async function main(ns) {  
  ns.print("Starting script here");  
  await ns.hack("foodnstuff"); //Use Netscript hack function  
  ns.print(ns.args);           //The script arguments must be prefaced with ns as_  
  ↪well  
}
```

Here is a summary of all rules you need to follow when writing Netscript JS code:

- Write `await` before any call to the following Netscript functions:
 - `hack`
 - `grow`
 - `weaken`
 - `sleep`
 - `prompt`
 - `wget`
 - `scp`
 - `write`
 - `writePort`
- Any function that contains `await` must be declared as `async`
- Always `await` any function that is marked as `async`
- Any functions that you want to be visible from other scripts must be marked with `export`.
- **Do not write any infinite loops without using a `sleep` or one of the timed Netscript functions like `hack`.** Doing so will freeze your game.
- Any global variable declared in a `ns2` script is shared between all instances of that script. For example, assume you write a script `foo.js` and declared a global variable like so:

```
//foo.js  
let globalVariable;  
  
export async function main(ns) {  
  globalVariable = ns.args.length;  
  while(true) {  
    ns.tprint(globalVariable);  
    await ns.sleep(3000);  
  }  
}
```

Then, you ran multiple instances of `foo.js`:

```
$ run foo.js 1  
$ run foo.js 1 2 3  
$ run foo.js 1 2 3 4 5
```

Then all three instances of `foo.js` will share the same instance of `globalVariable`. (In this example, the value of `globalVariable` will be set to 5 because the last instance of `foo.js` to run has 5 arguments. This means that all three instances of the script will repeatedly print the value 5).

These global variables can be thought of as [C++ static class members](#), where a ns2 script is a class and a global variable is a static member within that class.

1.1.3.3 Example

early-hack-template.script

```
var target = args[0];
var moneyThresh = getServerMaxMoney(target) * 0.75;
var securityThresh = getServerMinSecurityLevel(target) + 5;
if (fileExists("BruteSSH.exe", "home")) {
    brutessh(target);
}
nuke(target);
while(true) {
    if (getServerSecurityLevel(target) > securityThresh) {
        weaken(target);
    } else if (getServerMoneyAvailable(target) < moneyThresh) {
        grow(target);
    } else {
        hack(target);
    }
}
```

early-hack-template.js

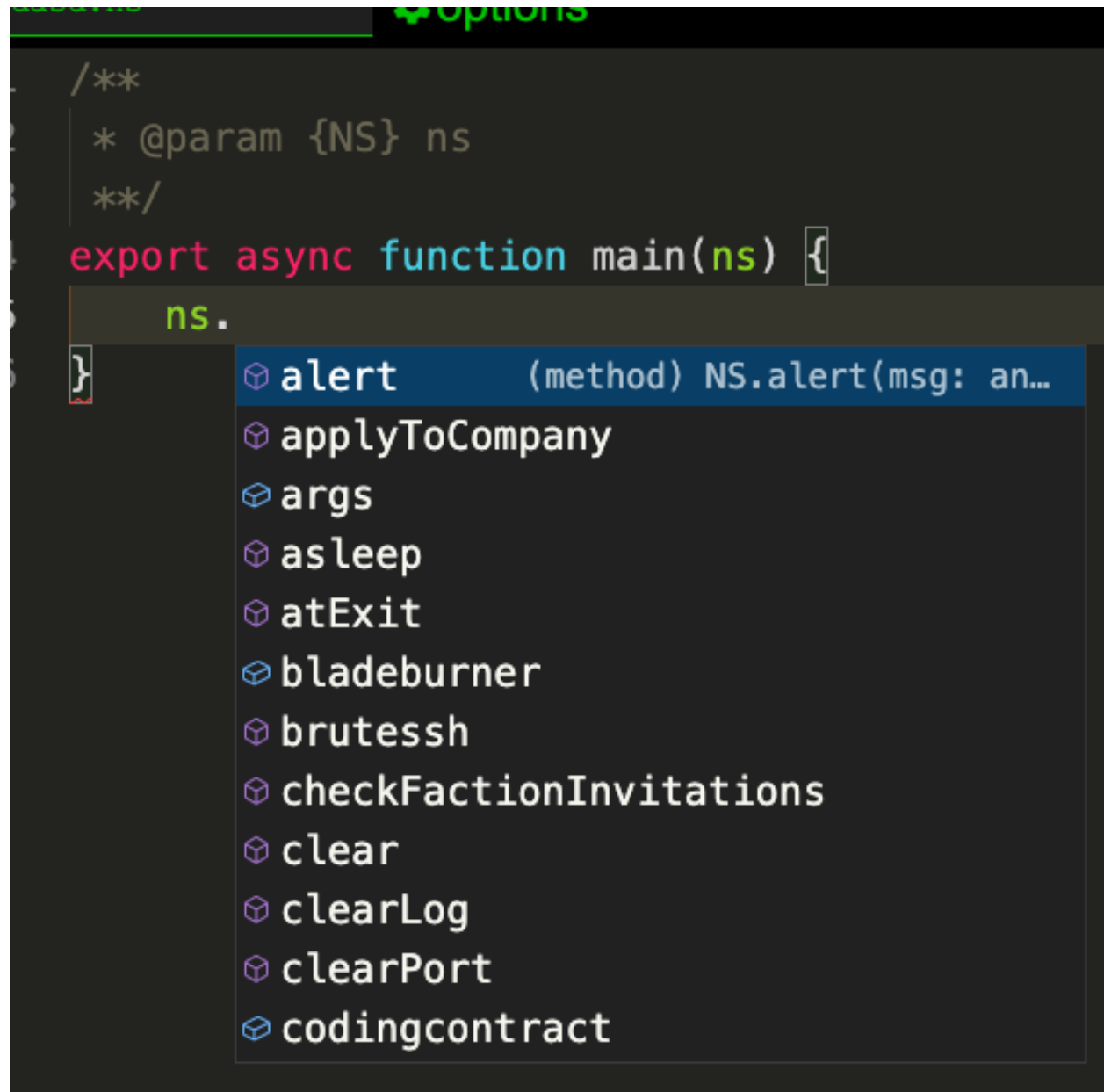
```
export async function main(ns) {
    var target = ns.args[0];
    var moneyThresh = ns.getServerMaxMoney(target) * 0.75;
    var securityThresh = ns.getServerMinSecurityLevel(target) + 5;
    if (ns.fileExists("BruteSSH.exe", "home")) {
        ns.brutessh(target);
    }
    ns.nuke(target);
    while(true) {
        if (ns.getServerSecurityLevel(target) > securityThresh) {
            await ns.weaken(target);
        } else if (ns.getServerMoneyAvailable(target) < moneyThresh) {
            await ns.grow(target);
        } else {
            await ns.hack(target);
        }
    }
}
```

1.1.3.4 What's with the weird comment

You may have noticed that every new ns2 file will contains the following comment.

```
/**
 * @param {NS} ns
 */
```

This comment is used to help the text editor autocomplete functions in the Netscript API. You can enable it by pressing ctrl+space after *ns*.



The comment can be safely removed but it is recommended to keep it as it will help you.

1.1.4 Netscript Script Arguments

Arguments passed into a script can be accessed in Netscript using a special array called *args*. The arguments can be accessed using a normal array using the `[]` operator (`args[0]`, `args[1]`, etc. ...).

For example, let's say we want to make a generic script 'generic-run.script' and we plan to pass two arguments into that script. The first argument will be the name of another script, and the second argument will be a number. This generic script will run the script specified in the first argument with the amount of threads specified in the second element. The code would look like:

```
run(args[0], args[1]);
```

It is also possible to get the number of arguments that was passed into a script using:

```
args.length
```

WARNING: Do not try to modify the args array. This will break the game.

example for accessing arguments in ns2 from terminal execution: terminal command: run name_of_script.js -t 10 -tail argument1 argument2

ns2 script:

```
const args_obj = arguments[0] const argument1 = (args_obj.server.args[0]) const argument2 = (args_obj.server.args[1])
```

1.1.5 Netscript Basic Functions

This page contains a subset of functions that are available in Bitburner. For the complete list see <https://github.com/danielyxie/bitburner/tree/dev/markdown> This includes information such as function signatures, what they do, and their return values.

1.1.5.1 hack() Netscript Function

hack (*hostname*[, *opts*={}])

RAM cost 0.1 GB

Arguments

- **hostname** (*string*) – Hostname of the target server.
- **opts** (*object*) – Optional parameters for configuring function behavior. Properties:
 - **threads** (*number*) - Number of threads to use for this function. Must be less than or equal to the number of threads the script is running with.
 - **stock** (*boolean*) - If true, the function can affect the stock market. See [Player Actions Influencing Stocks](#)

Returns The amount of money stolen if the hack is successful, and zero otherwise

Function that is used to try and hack servers to steal money and gain hacking experience. The runtime for this command depends on your hacking level and the target server's security level. In order to hack a server you must first gain root access to that server and also have the required hacking level.

A script can hack a server from anywhere. It does not need to be running on the same server to hack that server. For example, you can create a script that hacks the 'foodnstuff' server and run that script on any server in the game.

A successful [hack](#) on a server will raise that server's security level by 0.002.

Action time is calculated at the start, effect is calculated at the end.

Example:

```
hack("foodnstuff");
hack("10.1.2.3");
hack("foodnstuff", { threads: 5 }); // Only use 5 threads to hack
```

1.1.5.2 `grow()` Netscript Function

grow (*hostname* [, *opts*={}])

RAM cost 0.15 GB

Arguments

- **hostname** (*string*) – Hostname of the target server.
- **opts** (*object*) – Optional parameters for configuring function behavior. Properties:
 - **threads** (*number*) - Number of threads to use for this function. Must be less than or equal to the number of threads the script is running with.
 - **stock** (*boolean*) - If true, the function can affect the stock market. See [Player Actions Influencing Stocks](#)

Returns The number by which the money on the server was multiplied for the growth

Increase the amount of money available on a server. The time it takes to execute depends on your hacking level and the target server's security level. When `grow` completes, the money available on a target server will be increased by a certain, fixed percentage. This percentage is determined by the target server's growth rate (which varies between servers) and security level. Generally, higher-level servers have higher growth rates.

Like `hack`, `grow` can be called on any server, from any server. The `grow` command requires root access to the target server, but there is no required hacking level to run the command. It also raises the security level of the target server by 0.004 per thread.

Action time is calculated at the start, effect is calculated at the end.

Example:

```
while(true) {  
    grow("foodnstuff");  
}
```

1.1.5.3 `weaken()` Netscript Function

weaken (*hostname* [, *opts*={}])

RAM cost 0.15 GB

Arguments

- **hostname** (*string*) – Hostname of the target server to weaken.
- **opts** (*object*) – Optional parameters for configuring function behavior. Properties:
 - **threads** (*number*) - Number of threads to use for this function. Must be less than or equal to the number of threads the script is running with.

Returns The amount by which the target server's security level was decreased. This is equivalent to 0.05 multiplied by the number of script threads.

Use your hacking skills to attack a server's security, lowering the server's security level. The runtime for this command depends on your hacking level and the target server's security level. This function lowers the security level of the target server by 0.05.

Like `hack` and `grow`, `weaken` can be called on any server, regardless of where the script is running. This command requires root access to the target server, but there is no required hacking level to run the command.

Example:

```
weaken("foodnstuff");  
weaken("foodnstuff", { threads: 5 }); // Only use 5 threads to weaken
```

1.1.5.4 sleep() Netscript Function

sleep (*n*)

RAM cost 0 GB

Arguments

- **n** (*number*) – Number of milliseconds to sleep

Suspends the script for *n* milliseconds.

Example:

```
sleep(3000); // Will wait 3 seconds.
```

1.1.5.5 print() Netscript Function

print (*args...*)

RAM cost 0 GB

Arguments

- **args** – Values to be printed.

Prints any number of values to the script's logs.

Example:

```
print("Hello world!"); // Prints "Hello world!" in the logs.  
print({a:5}); // Prints '{"a":5}' in the logs.
```

1.1.5.6 tprint() Netscript Function

tprint (*args...*)

RAM cost 0 GB

Arguments

- **args** – Values to be printed

Prints any number of values to the Terminal.

Example:

```
tprint("Hello world!"); // Prints "Hello world!" to the terminal.  
tprint({a:5}); // Prints '{"a":5}' to the terminal.
```

1.1.5.7 disableLog() Netscript Function

disableLog (*functionName*)

RAM cost 0 GB

Arguments

- **functionName** (*string*) – Name of function for which to disable logging.

Disables logging for the given function. Logging can be disabled for all functions by passing 'ALL' as the argument.

1.1.5.8 enableLog() Netscript Function

enableLog (*functionName*)

RAM cost 0 GB

Arguments

- **functionName** (*string*) – Name of function for which to enable logging.

Re-enables logging for the given function. If 'ALL' is passed into this function as an argument, then it will revert the effects of `disableLog('ALL')`

1.1.5.9 isLogEnabled() Netscript Function

isLogEnabled (*functionName*)

RAM cost 0 GB

Arguments

- **functionName** (*string*) – Name of function to check.

Returns `true` if logs are enabled for this function or for 'ALL'

Example:

```
isLogEnabled('hack'); // returns: true
```

1.1.5.10 scan() Netscript Function

scan (*hostname=current hostname*)

RAM cost 0.2 GB

Arguments

- **hostname** (*string*) – Hostname of the server to scan.

Returns array of strings of all the host directly connected to the target server.

Example:

```
scan("home"); // returns: ["foodnstuff", "sigma-cosmetics", "joesguns", "hong-  
→fang-tea", "harakiri-sushi", "iron-gym"]
```

1.1.5.11 nuke() Netscript Function

nuke (*hostname*)

RAM cost 0.05 GB

Arguments

- **hostname** (*string*) – Hostname of the target server.

Runs the `NUKE.exe` program on the target server. `NUKE.exe` must exist on your home computer.

Example:

```
nuke("foodnstuff");
```

1.1.5.12 brutessh() Netscript Function

brutessh (*hostname*)

RAM cost 0.05 GB

Arguments

- **hostname** (*string*) – Hostname of the target server.

Runs the `BruteSSH.exe` program on the target server. `BruteSSH.exe` must exist on your home computer.

Examples:

```
brutessh("foodnstuff");
```

1.1.5.13 ftpcrack() Netscript Function

ftpcrack (*hostname*)

RAM cost 0.05 GB

Arguments

- **hostname** (*string*) – Hostname of the target server.

Runs the `FTPCrack.exe` program on the target server. `FTPCrack.exe` must exist on your home computer.

Examples:

```
ftpcrack("foodnstuff");
```

1.1.5.14 relaysmtp() Netscript Function

relaysmtp (*hostname*)

RAM cost 0.05 GB

Arguments

- **hostname** (*string*) – Hostname of the target server.

Runs the `relaySMTP.exe` program on the target server. `relaySMTP.exe` must exist on your home computer.

Example:

```
relaysmtp("foodnstuff");
```

1.1.5.15 httpworm() Netscript Function

httpworm (*hostname*)

RAM cost 0.05 GB

Arguments

- **hostname** (*string*) – Hostname of the target server.

Runs the `HTTPWorm.exe` program on the target server. `HTTPWorm.exe` must exist on your home computer.

Example:

```
httpworm("foodnstuff");
```

1.1.5.16 sqlinject() Netscript Function

sqlinject (*hostname*)

RAM cost 0.05 GB

Arguments

- **hostname** (*string*) – Hostname of the target server.

Runs the `SQLInject.exe` program on the target server. `SQLInject.exe` must exist on your home computer.

Example:

```
sqlinject("foodnstuff");
```

1.1.5.17 run() Netscript Function

run (*script* [, *numThreads* = 1 [, *args...*]])

RAM cost 1 GB

Arguments

- **script** (*string*) – Filename of script to run
- **numThreads** (*number*) – Optional thread count for new script. Set to 1 by default. Will be rounded to nearest integer.
- **args...** – Additional arguments to pass into the new script that is being run. Note that if any arguments are being passed into the new script, then the second argument `numThreads` must be filled in with a value.

Returns The process id of the new process or 0 on failure.

Run a script as a separate process. This function can only be used to run scripts located on the current server (the server running the script that calls this function).

Warning: Running this function with a `numThreads` argument of 0 or less will cause a runtime error.

The simplest way to use the `run` command is to call it with just the script name. The following example will run `foo.script` single-threaded with no arguments:

```
run("foo.script");
```

The following example will run 'foo.script' but with 5 threads instead of single-threaded:

```
run("foo.script", 5);
```

This next example will run `foo.script` single-threaded, and will pass the string `foodnstuff` into the script as an argument:

```
run("foo.script", 1, 'foodnstuff');
```

1.1.5.18 exec() Netscript Function

exec (*script*, *hostname*[, *numThreads*=1[, *args...*]])

RAM cost 1.3 GB

Arguments

- **script** (*string*) – Filename of script to execute.
- **hostname** (*string*) – Hostname of the target server on which to execute the script.
- **numThreads** (*number*) – Optional thread count for new script. Set to 1 by default. Will be rounded to nearest integer
- **args...** – Additional arguments to pass into the new script that is being run. Note that if any arguments are being passed into the new script, then the third argument `numThreads` must be filled in with a value.

Returns Newly created process id on success, 0 on failure.

Run a script as a separate process on a specified server. This is similar to the `run` function except that it can be used to run a script on any server, instead of just the current server.

Warning: Running this function with a `numThreads` argument of 0 or less will cause a runtime error.

The simplest way to use the `exec` command is to call it with just the script name and the target server. The following example will try to run `generic-hack.script` on the `foodnstuff` server:

```
exec("generic-hack.script", "foodnstuff");
```

The following example will try to run the script `generic-hack.script` on the `joesguns` server with 10 threads:

```
exec("generic-hack.script", "joesguns", 10);
```

This last example will try to run the script `foo.script` on the `foodnstuff` server with 5 threads. It will also pass the number 1 and the string “test” in as arguments to the script:

```
exec("foo.script", "foodnstuff", 5, 1, "test");
```

1.1.5.19 spawn() Netscript Function

spawn (*script*, *numThreads*[, *args...*])

RAM cost 2 GB

Arguments

- **script** (*string*) – Filename of script to execute
- **numThreads** (*number*) – Number of threads to spawn new script with. Will be rounded to nearest integer.
- **args...** – Additional arguments to pass into the new script that is being run.

Terminates the current script, and then after a delay of about 10 seconds it will execute the newly-specified script. The purpose of this function is to execute a new script without being constrained by the RAM usage of the current one. This function can only be used to run scripts on the local server.

Warning: Running this function with a `numThreads` argument of 0 or less will cause a runtime error.

Example:

```
spawn('foo.script', 10, 'foodnstuff', 90); // "run foo.script foodnstuff 90 -t 10  
→" in 10 seconds.
```

1.1.5.20 kill() Netscript Function

kill (*script*, *hostname*[, *args...*])

RAM cost 0.5 GB

Arguments

- **script** (*string*) – Filename of the script to kill.
- **hostname** (*string*) – Hostname of the server on which to kill the script.
- **args...** – Arguments to identify which script to kill.

Returns `true` is that script was killed.

Kills the script on the target server specified by the script’s name and arguments. Remember that scripts are uniquely identified by both their name and arguments. For example, if `foo.script` is run with the argument 1, then this is not the same as `foo.script` run with the argument 2, even though they have the same code.

Examples:

The following example will try to kill a script named `foo.script` on the `foodnstuff` server that was ran with no arguments:

```
kill("foo.script", "foodnstuff");
```

The following will try to kill a script named `foo.script` on the current server that was ran with no arguments:

```
kill("foo.script", getHostname());
```

The following will try to kill a script named `foo.script` on the current server that was ran with the arguments 1 and “foodnstuff”:

```
kill("foo.script", getHostname(), 1, "foodnstuff");
```

kill (*scriptPid*)

RAM cost 0.5 GB

Arguments

- **scriptPid** (*number*) – PID of the script to kill

Returns `true` that script was killed.

Kills the script with the specified PID. Killing a script by its PID will typically have better performance, especially if you have many scripts running.

Example:

```
if (kill(10)) {
    print("Killed script with PID 10!");
}
```

1.1.5.21 killall() Netscript Function

killall (*hostname*)

RAM cost 0.5 GB

Arguments

- **hostname** (*string*) – Hostname of the server on which to kill all scripts.

Returns `true` if scripts were killed on target server.

Kills all running scripts on the specified server.

Example:

```
killall('foodnstuff'); // returns: true
```

1.1.5.22 scp() Netscript Function

scp (*files* [, *source*], *destination*)

RAM cost 0.6 GB

Arguments

- **files** (*string/array*) – Filename or an array of filenames of script/literature files to copy
- **source** (*string*) – Hostname of the source server, which is the server from which the file will be copied. This argument is optional and if it’s omitted the source will be the current server.

- **destination** (*string*) – Hostname of the destination server, which is the server to which the file will be copied.

Returns `true` if the copy was a success.

Copies a script or literature (.lit) file(s) to another server. The `files` argument can be either a string specifying a single file to copy, or an array of strings specifying multiple files to copy.

If the `files` argument is an array then this function will return `true` if at least one of the files in the array is successfully copied.

Example:

```
//Copies "hack-template.script" from the current server to "foodnstuff"
scp("hack-template.script", "foodnstuff"); // returns: true

//Copies "foo.lit" from the helios server to the "home" computer
scp("foo.lit", "helios", "home"); // returns: true

//Tries to copy three files from "rothman-uni" to "home" computer
files = ["fool.lit", "foo2.script", "foo3.script"];
scp(files, "rothman-uni", "home"); // returns: true
```

1.1.5.23 ls() Netscript Function

ls (*hostname* [, *grep*])

RAM cost 0.2 GB

Arguments

- **hostname** (*string*) – Hostname of the target server.
- **grep** (*string*) – a substring to search for in the filename.

Returns String array of all files in alphabetical order.

Example:

```
ls("home"); // returns: ["demo.script", "msg1.txt"]
```

1.1.5.24 ps() Netscript Function

ps ([*hostname=current hostname*])

RAM cost 0.2 GB

Arguments

- **hostname** (*string*) – Hostname address of the target server. If not specified, it will be the current server's IP by default.

Returns array of object

Returns an array with general information about all scripts running on the specified target server. The information for each server is given in an object with the following structure:

```
{
  filename: Script name,
  threads:  Number of threads script is running with,
  args:      Script's arguments,
  pid:       Script's pid
}
```

Example:

```
processes = ps("home");
for (let i = 0; i < processes.length; ++i) {
  tprint(processes[i].filename + ' ' + processes[i].threads);
  tprint(processes[i].args);
  tprint(processes[i].pid);
}
```

1.1.5.25 hasRootAccess() Netscript Function

hasRootAccess (*hostname*)

RAM cost 0.05 GB

Arguments

- **hostname** (*string*) – Hostname of the target server.

Returns `true` if you have root access on the target server.

Example:

```
if (hasRootAccess("foodnstuff") == false) {
  nuke("foodnstuff");
}
```

1.1.5.26 getHackingLevel() Netscript Function

getHackingLevel ()

RAM cost 0.05 GB

Returns The player's current hacking level.

Example:

```
getHackingLevel(); // returns: 124
```

1.1.5.27 getHackingMultipliers() Netscript Function

getHackingMultipliers ()

RAM cost 4 GB

Returns object containing the player's hacking multipliers. These multipliers are returned in decimal forms, not percentages (e.g. 1.5 instead of 150%).

Structure:

```
{
  chance: Player's hacking chance multiplier,
  speed: Player's hacking speed multiplier,
  money: Player's hacking money stolen multiplier,
  growth: Player's hacking growth multiplier
}
```

Example:

```
mults = getHackingMultipliers();
print(mults.chance);
print(mults.growth);
```

1.1.5.28 getHacknetMultipliers() Netscript Function

getHacknetMultipliers()

RAM cost 4 GB

Returns object containing the player's hacknet multipliers. These multipliers are returned in decimal forms, not percentages (e.g. 1.5 instead of 150%).

Structure:

```
{
  production: Player's hacknet production multiplier,
  purchaseCost: Player's hacknet purchase cost multiplier,
  ramCost: Player's hacknet ram cost multiplier,
  coreCost: Player's hacknet core cost multiplier,
  levelCost: Player's hacknet level cost multiplier
}
```

Example:

```
mults = getHacknetMultipliers();
print(mults.production);
print(mults.purchaseCost);
```

1.1.5.29 getServerMoneyAvailable() Netscript Function

getServerMoneyAvailable (*hostname*)

RAM cost 0.1 GB

Arguments

- **hostname** (*string*) – Hostname of target server.

Returns Money available on that server.

Note: Running this function on the home computer will return the player's money.

Example:

```
getServerMoneyAvailable("foodnstuff"); // returns: 120000
getServerMoneyAvailable("home"); // returns: 1000
```

1.1.5.30 `getServerMaxMoney()` Netscript Function

getServerMaxMoney (*hostname*)

RAM cost 0.1 GB

Arguments

- **hostname** (*string*) – Hostname of target server.

Returns Maximum amount of money that can be available on a server.

Example:

```
getServerMaxMoney('foodnstuff'); // returns: 50000000
```

1.1.5.31 `getServerSecurityLevel()` Netscript Function

getServerSecurityLevel (*hostname*)

RAM cost 0.1 GB

Arguments

- **hostname** (*string*) – Hostname of target server.

Returns The security level of the target server.

Example:

```
getServerSecurityLevel("foodnstuff"); // returns: 3.45
```

1.1.5.32 `getServerMinSecurityLevel()` Netscript Function

getServerMinSecurityLevel (*hostname*)

RAM cost 0.1 GB

Arguments

- **hostname** (*string*) – Hostname of target server.

Returns The minimum security level of the target server.

Example:

```
getServerMinSecurityLevel('foodnstuff'); // returns: 3
```

1.1.5.33 `getServerRequiredHackingLevel()` Netscript Function

getServerRequiredHackingLevel (*hostname*)

RAM cost 0.1 GB

Arguments

- **hostname** (*string*) – Hostname of target server.

Returns The required hacking level of target server.

Example:

```
getServerRequiredHackingLevel("foodnstuff"); // returns: 5
```

1.1.5.34 getServerNumPortsRequired() Netscript Function

getServerNumPortsRequired (*hostname*)

RAM cost 0.1 GB

Arguments

- **hostname** (*string*) – Hostname of target server.

Returns The number of open ports required to successfully run NUKE.exe on the specified server.

Example:

```
getServerNumPortsRequired("unitalife"); // returns: 4
```

1.1.5.35 getServerMaxRam() Netscript Function

getServerMaxRam (*hostname*)

RAM cost 0.05 GB

Arguments

- **hostname** (*string*) – Hostname of target server.

Returns Total ram available on that server. In GB.

Example:

```
maxRam = getServerMaxRam("helios"); // returns: 16
print("helios has "+maxRam + "GB");
```

1.1.5.36 getServerUsedRam() Netscript Function

getServerUsedRam (*hostname*)

RAM cost 0.05 GB

Arguments

- **hostname** (*string*) – Hostname of target server.

Returns Used ram on that server. In GB.

Example:

```
usedRam = getServerUsedRam("harakiri-sushi"); // returns: 5.6
print("harakiri-sushi uses "+usedRam + "GB");
```

1.1.5.37 serverExists() Netscript Function

serverExists (*hostname*)

RAM cost 0.1 GB

Arguments

- **hostname** (*string*) – Hostname of target server.

Returns `true` if the target server exists.

Example:

```
serverExists("foodnstuff"); // returns: true
```

1.1.5.38 fileExists() Netscript Function

fileExists (*filename* [, *hostname*])

RAM cost 0.1 GB

Arguments

- **filename** (*string*) – Filename of file to check.
- **hostname** (*string*) – Hostname of target server. This is optional. If it is not specified then the function will use the current server as the target server.

Returns `true` if the file exists, `false` if it doesn't.

The filename for scripts is case-sensitive, but for other types of files it is not. For example, `fileExists("brutessh.exe")` will work fine, even though the actual program is named `BruteSSH.exe`.

If the `hostname` argument is omitted, then the function will search through the server running the script that calls this function for the file.

Examples:

```
fileExists("foo.script", "foodnstuff"); // returns: false
fileExists("ftpcrack.exe"); // returns: true
```

The first example above will return `true` if the script named `foo.script` exists on the `foodnstuff` server, and `false` otherwise. The second example above will return `true` if the current server contains the `FTPCrack.exe` program, and `false` otherwise.

1.1.5.39 isRunning() Netscript Function

isRunning (*filename* [, *hostname*=current hostname [, *args...*]])

RAM cost 0.1 GB

Arguments

- **filename** (*string*) – Filename of script to check. case-sensitive.
- **hostname** (*string*) – Hostname of target server. Defaults to current server
- **args...** – Arguments to specify/identify which scripts to search for

Returns `true` if that script with those args is running on that server.

Note: Remember that a script is uniquely identified by both its name and its arguments.

Examples:

In this first example below, the function call will return true if there is a script named `foo.script` with no arguments running on the `foodnstuff` server, and false otherwise:

```
isRunning("foo.script", "foodnstuff");
```

In this second example below, the function call will return true if there is a script named `foo.script` with no arguments running on the current server, and false otherwise:

```
isRunning("foo.script", getHostname());
```

In this next example below, the function call will return true if there is a script named `foo.script` running with the arguments 1, 5, and “test” (in that order) on the `joesguns` server, and false otherwise:

```
isRunning("foo.script", "joesguns", 1, 5, "test");
```

isRunning (*scriptPid*)

RAM cost 0.1 GB

Arguments

- **scriptPid** (*number*) – PID of the script to check.

Same as the above version but with pid.

Example:

```
isRunning(39);
```

1.1.5.40 getPurchasedServerCost() Netscript Function

getPurchasedServerCost (*ram*)

RAM cost 0.25 GB

Arguments

- **ram** (*number*) – Amount of RAM of a potential purchased server. Must be a power of 2 (2, 4, 8, 16, etc.). Maximum value of 1048576 (2²⁰)

Returns Cost to purchase a server with the specified amount of `ram`.

Example:

```
getPurchasedServerCost(8192); // returns: 450560000
```

1.1.5.41 purchaseServer() Netscript Function

purchaseServer (*hostname, ram*)

RAM cost 2.25 GB

Arguments

- **hostname** (*string*) – Hostname of the purchased server.
- **ram** (*number*) – Amount of RAM of the purchased server. Must be a power of 2. Maximum value of `getPurchasedServerMaxRam`

Returns The hostname of the newly purchased server. Empty string on failure.

Purchased a server with the specified hostname and amount of RAM.

The `hostname` argument can be any data type, but it will be converted to a string and have whitespace removed. Anything that resolves to an empty string will cause the function to fail. If there is already a server with the specified hostname, then the function will automatically append a number at the end of the `hostname` argument value until it finds a unique hostname. For example, if the script calls `purchaseServer("foo", 4)` but a server named “foo” already exists, then it will automatically change the hostname to “foo-0”. If there is already a server with the hostname “foo-0”, then it will change the hostname to “foo-1”, and so on.

Note that there is a maximum limit to the amount of servers you can purchase.

Example:

```
ram = 64;
hn = "pserv-";
for (i = 0; i < 5; ++i) {
    purchaseServer(hn + i, ram);
}
```

1.1.5.42 deleteServer() Netscript Function

deleteServer (*hostname*)

RAM cost 2.25 GB

Arguments

- **hostname** (*string*) – Hostname of the server to delete.

Returns `true` if successful, `false` otherwise.

Deletes the specified purchased server.

The `hostname` argument can be any data type, but it will be converted to a string. Whitespace is automatically removed from the string. This function will not delete a server that still has scripts running on it.

1.1.5.43 getPurchasedServers() Netscript Function

getPurchasedServers ()

RAM cost 2.25 GB

Returns String array of hostnames of all of the servers you have purchased.

Example:

```
getPurchasedServers(); // returns: ['grow-server-0', 'grow-server-1', 'weaken-
↪server-0']
```

1.1.5.44 getPurchasedServerLimit() Netscript Function

getPurchasedServerLimit ()

RAM cost 0.05 GB

Returns The maximum number of servers you can purchase.

Example:

```
getPurchasedServerLimit() // returns: 25
```

1.1.5.45 getPurchasedServerMaxRam() Netscript Function

getPurchasedServerMaxRam()

RAM cost 0.05 GB

Returns The maximum RAM that a purchased server can have.

Example:

```
getPurchasedServerMaxRam(); // returns: 1048576
```

1.1.5.46 scriptRunning() Netscript Function

scriptRunning (*scriptname*, *hostname*)

RAM cost 1 GB

Arguments

- **scriptname** (*string*) – Filename of script to check. case-sensitive.
- **hostname** (*string*) – Hostname of target server.

Returns `true` if any script with that file name is running on that server.

This is different than the *isRunning* function because it does not try to identify a specific instance of a running script by its arguments.

Examples:

The example below will return true if there is any script named `foo.script` running on the `foodnstuff` server, and false otherwise:

```
scriptRunning("foo.script", "foodnstuff");
```

The example below will return true if there is any script named `foo.script` running on the current server, and false otherwise:

```
scriptRunning("foo.script", getHostname());
```

1.1.5.47 scriptKill() Netscript Function

scriptKill (*scriptname*, *hostname*)

RAM cost 1 GB

Arguments

- **scriptname** (*string*) – Filename of script to kill. case-sensitive.
- **hostname** (*string*) – Hostname of target server.

Returns `true` if any scripts were killed.

Kills all scripts with the specified filename on the target server specified by `hostname`, regardless of arguments.

Example:

```
scriptKill("demo.script", "home"); // returns: true
```

1.1.5.48 getScriptRam() Netscript Function

getScriptRam (*filename* [, *hostname*])

RAM cost 0.1 GB

Arguments

- **filename** (*string*) – Filename of script.
- **hostname** (*string*) – Hostname of target server the script is located on. Default to the server this script is running on.

Returns Amount of RAM required to run the script, 0 if it does not exist.

Example:

```
getScriptRam("grow.script"); // returns: 1.75
```

1.1.6 Netscript Advanced Functions

These Netscript functions become relevant later on in the game. They are put on a separate page because they contain spoilers for the game.

Warning: This page contains spoilers for the game

1.1.6.1 autocomplete() Netscript Function

Warning: This feature is not officially supported yet and the API might change. It is also only supported in ns2

autocomplete (*data*, *args*)

RAM cost 0 GB

Arguments

- **data** (*Object*) – general data about the game you might want to autocomplete.
- **args** (*string[]*) – current arguments. Minus *run script.js*

data is an object with the following properties:

```
{
  servers: list of all servers in the game.
  txts:    list of all text files on the current server.
  scripts: list of all scripts on the current server.
  flags:   the same flags function as passed with ns. Calling this function,
  ↪ adds all the flags as autocomplete arguments
}
```

This function is special as it must be declared as a top level function like *main*.

Example:

```
export function autocomplete(data, args) {  
  return [...data.servers]; // This script autocompletes the list of servers.  
  return [...data.servers, ...data.scripts]; // Autocomplete servers and scripts  
  return ["low", "medium", "high"]; // Autocomplete 3 specific strings.  
}
```

Terminal:

```
$ run demo.js mega\t  
// results in  
$ run demo.js megacorp
```

1.1.6.2 Injecting HTML in the game

Bitburner uses React and Material-UI to render everything. Modifying the UI is possible but not officially supported.

To automatically enter commands in the terminal (only works if looking at the terminal):

```
// Acquire a reference to the terminal text field  
const terminalInput = document.getElementById("terminal-input");  
  
// Set the value to the command you want to run.  
terminalInput.value="home;connect n00dles;home;connect n00dles;home;";  
  
// Get a reference to the React event handler.  
const handler = Object.keys(terminalInput)[1];  
  
// Perform an onChange event to set some internal values.  
terminalInput[handler].onChange({target:terminalInput});  
  
// Simulate an enter press  
terminalInput[handler].onKeyDown({key:'Enter',preventDefault:()=>null});
```

To add lines to the terminal (only works if looking at the terminal):

```
// Acquire a reference to the terminal list of lines.  
const list = document.getElementById("generic-react-container").querySelector("ul");  
  
// Inject some HTML.  
list.insertAdjacentHTML('beforeend','<li><p color=lime>whatever custom html</p></li>');
```

1.1.7 Netscript Hacknet Node API

Warning: Not all functions in the Hacknet Node API are immediately available. For this reason, the documentation for this API may contain spoilers for the game.

Netscript provides the following API for accessing and upgrading your Hacknet Nodes through scripts.

Note that none of these functions will write to the script's logs. If you want to see what your script is doing you will have to print to the logs yourself.

Hacknet Node API functions must be accessed through the hacknet namespace

In Netscript 1.0:

```
hacknet.purchaseNode();
hacknet.getNodeStats(3).level;
```

In *NS2*:

```
ns.hacknet.purchaseNode();
ns.hacknet.getNodeStats(3).level;
```

1.1.7.1 numNodes() Netscript Function

numNodes()

RAM cost 0 GB

Returns Number of Hacknet Nodes you own.

1.1.7.2 purchaseNode() Netscript Function

purchaseNode()

RAM cost 0 GB

Returns Index of the newly purchased node. -1 on failure.

Purchases a new Hacknet Node. This index is equivalent to the number at the end of the Hacknet Node's name (e.g The Hacknet Node named 'hacknet-node-4' will have an index of 4).

1.1.7.3 getPurchaseNodeCost() Netscript Function

getPurchaseNodeCost()

RAM cost 0 GB

Returns Cost of purchasing a new Hacknet Node.

1.1.7.4 getNodeStats() Netscript Function

Warning: This page contains spoilers for the game

getNodeStats(*i*)

RAM cost 0 GB

Arguments

- **i** (*number*) – Index of Hacknet Node. *See [here for details](#)*

Returns Object containing a variety of stats about the specified Hacknet Node

```
{
  name:           Node's name ("hacknet-node-5"),
  level:          Node's level,
  ram:            Node's RAM,
  cores:          Node's number of cores,
  cache:          Cache level. Only applicable for Hacknet Servers
  hashCapacity:   Hash Capacity provided by this Node. Only applicable for
↪ Hacknet Servers
  production:     Node's production per second
  timeOnline:     Number of seconds since Node has been purchased,
  totalProduction: Total amount that the Node has produced
}
```

Note: Note that for Hacknet Nodes, production refers to the amount of money the node generates. For Hacknet Servers (the upgraded version of Hacknet Nodes), production refers to the amount of hashes the node generates.

1.1.7.5 upgradeLevel() Netscript Function

upgradeLevel (*i*, *n*)

RAM cost 0 GB

Arguments

- **i** (*number*) – Index of Hacknet Node. *See here for details*
- **n** (*number*) – Number of levels to purchase. Must be positive. Rounded to nearest integer

Returns `true` if the upgrade was successful.

Tries to upgrade the level of the specified Hacknet Node by *n*.

1.1.7.6 upgradeRam() Netscript Function

upgradeRam (*i*, *n*)

RAM cost 0 GB

Arguments

- **i** (*number*) – Index of Hacknet Node. *See here for details*
- **n** (*number*) – Number of times to upgrade RAM. Must be positive. Rounded to nearest integer.

Returns `true` if the upgrade was successful.

Tries to upgrade the specified Hacknet Node's RAM *n* times. Note that each upgrade doubles the Node's RAM. So this is equivalent to multiplying the Node's RAM by 2^n .

1.1.7.7 upgradeCore() Netscript Function

upgradeCore (*i*, *n*)

RAM cost 0 GB

Arguments

- **i** (*number*) – Index of Hacknet Node. [See here for details](#)
- **n** (*number*) – Number of cores to purchase. Must be positive. Rounded to nearest integer

Returns `true` if the upgrade was successful.

Tries to purchase `n` cores for the specified Hacknet Node.

1.1.7.8 `getLevelUpgradeCost()` Netscript Function

`getLevelUpgradeCost` (*i*, *n*)

RAM cost 0 GB

Arguments

- **i** (*number*) – Index of Hacknet Node. [See here for details](#)
- **n** (*number*) – Number of levels to upgrade. Must be positive. Rounded to nearest integer

Returns Cost of upgrading the specified Hacknet Node by `n` levels.

If an invalid value for `n` is provided, then this function returns 0. If the specified Hacknet Node is already at max level, then `Infinity` is returned.

1.1.7.9 `getRamUpgradeCost()` Netscript Function

`getRamUpgradeCost` (*i*, *n*)

RAM cost 0 GB

Arguments

- **i** (*number*) – Index of Hacknet Node. [See here for details](#)
- **n** (*number*) – Number of times to upgrade RAM. Must be positive. Rounded to nearest integer.

Returns Cost of upgrading the RAM of the specified Hacknet Node `n` times.

If an invalid value for `n` is provided, then this function returns 0. If the specified Hacknet Node is already at max RAM, then `Infinity` is returned.

1.1.7.10 `getCoreUpgradeCost()` Netscript Function

`getCoreUpgradeCost` (*i*, *n*)

RAM cost 0 GB

Arguments

- **i** (*number*) – Index of Hacknet Node. [See here for details](#)
- **n** (*number*) – Number of times to upgrade cores. Must be positive. Rounded to nearest integer

Returns Cost of upgrading the number of cores of the specified Hacknet Node by `n`.

If an invalid value for `n` is provided, then this function returns 0. If the specified Hacknet Node is already at the max number of cores, then `Infinity` is returned.

1.1.7.11 Referencing a Hacknet Node

Most of the functions in the Hacknet Node API perform an operation on a single Node. Therefore, a numeric index is used to identify and specify which Hacknet Node a function should act on. This index number corresponds to the number at the end of the name of the Hacknet Node. For example, the first Hacknet Node you purchase will have the name “hacknet-node-0” and is referenced using index 0. The fifth Hacknet Node you purchase will have the name “hacknet-node-4” and is referenced using index 4.

1.1.7.12 RAM Cost

Accessing the *hacknet* namespace incurs a one time cost of 4 GB of RAM. In other words, using multiple Hacknet Node API functions in a script will not cost more than 4 GB of RAM.

1.1.7.13 Utilities

The following functions are not officially part of the Hacknet Node API, but they can be useful when writing Hacknet Node-related scripts. Since they are not part of the API, they do not need to be accessed using the *hacknet* namespace.

- `getHacknetMultipliers()`

1.1.7.14 Example(s)

The following is an example of one way a script can be used to automate the purchasing and upgrading of Hacknet Nodes.

This script attempts to purchase Hacknet Nodes until the player has a total of 8. Then it gradually upgrades those Node’s to a minimum of level 80, 16 GB RAM, and 8 cores

```
function myMoney() {
    return getServerMoneyAvailable("home");
}

disableLog("getServerMoneyAvailable");
disableLog("sleep");

var cnt = 8;

while(hacknet.numNodes() < cnt) {
    res = hacknet.purchaseNode();
    print("Purchased hacknet Node with index " + res);
};

for (var i = 0; i < cnt; i++) {
    while (hacknet.getNodeStats(i).level <= 80) {
        var cost = hacknet.getLevelUpgradeCost(i, 10);
        while (myMoney() < cost) {
            print("Need $" + cost + " . Have $" + myMoney());
            sleep(3000);
        }
        res = hacknet.upgradeLevel(i, 10);
    }
};

print("All nodes upgraded to level 80");
```

(continues on next page)

(continued from previous page)

```

for (var i = 0; i < cnt; i++) {
    while (hacknet.getNodeStats(i).ram < 16) {
        var cost = hacknet.getRamUpgradeCost(i, 2);
        while (myMoney() < cost) {
            print("Need $" + cost + " . Have $" + myMoney());
            sleep(3000);
        }
        res = hacknet.upgradeRam(i, 2);
    };
};

print("All nodes upgraded to 16GB RAM");

for (var i = 0; i < cnt; i++) {
    while (hacknet.getNodeStats(i).cores < 8) {
        var cost = hacknet.getCoreUpgradeCost(i, 1);
        while (myMoney() < cost) {
            print("Need $" + cost + " . Have $" + myMoney());
            sleep(3000);
        }
        res = hacknet.upgradeCore(i, 1);
    };
};

print("All nodes upgraded to 8 cores");

```

1.1.8 Netscript Miscellaneous

1.1.8.1 Netscript Ports

Netscript Ports are endpoints that can be used to communicate between scripts. A port is implemented as a sort of serialized queue, where you can only write and read one element at a time from the port. When you read data from a port, the element that is read is removed from the port.

The `read()`, `write()`, `tryWrite()`, `clear()`, and `peek()` Netscript functions can be used to interact with ports.

Right now, there are only 20 ports for Netscript, denoted by the number 1 through 20. When using the functions above, the ports are specified by passing the number as the first argument.

IMPORTANT: The data inside ports are not saved! This means if you close and re-open the game, or reload the page then you will lose all of the data in the ports!

Example Usage

Here's a brief example of how ports work. For the sake of simplicity we'll only deal with port 1.

Let's assume Port 1 starts out empty (no data inside). We'll represent the port as such:

```
[ ]
```

Now assume we ran the following simple script:

```

for (i = 0; i < 10; ++i) {
    writePort(1, i); //Writes the value of i to port 1
}

```

After this script executes, our script will contain every number from 0 through 9, as so:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Then, assume we run the following script:

```
for (i = 0; i < 3; ++i) {  
    print(readPort(1)); //Reads a value from port 1 and then prints it  
}
```

This script above will read the first three values from port 1 and then print them to the script's log. The log will end up looking like:

```
0  
1  
2
```

And the data in port 1 will look like:

```
[3, 4, 5, 6, 7, 8, 9]
```

Warning: In *NS2*, do not try writing base *Promises* to a port.

Port Handles

WARNING: Port Handles only work in *NS2*. They do not work in *Netscript 1.0*

The `getPortHandle()` Netscript function can be used to get a handle to a Netscript Port. This handle allows you to access several new port-related functions. The functions are:

`NetscriptPort.writePort(data)`

Arguments

- **data** – Data to write to the port

Returns If the port is full, the item that is removed from the port is returned. Otherwise, null is returned.

Writes *data* to the port. Works the same as the Netscript function *write*.

`NetscriptPort.tryWritePort(data)`

Arguments

- **data** – Data to try to write to the port

Returns True if the data is successfully written to the port, and false otherwise.

Attempts to write *data* to the Netscript port. If the port is full, the data will not be written. Otherwise, the data will be written normally.

`NetscriptPort.full()`

Returns True if the Netscript Port is full, and false otherwise

`NetscriptPort.empty()`

Returns True if the Netscript Port is empty, and false otherwise

`NetscriptPort.clear()`

Clears all data from the port. Works the same as the Netscript function *clear*

Port Handle Example:

```
port = getPortHandle(5);
back = port.data.pop(); //Get and remove last element in port

//Wait for port data before reading
while(port.empty()) {
    sleep(10000);
}
res = port.read();

//Wait for there to be room in a port before writing
while (!port.tryWrite(5)) {
    sleep(5000);
}

//Successfully wrote to port!
```

1.1.8.2 Comments

Netscript supports comments using the same syntax as Javascript comments. Comments are not evaluated as code, and can be used to document and/or explain code:

```
//This is a comment and will not get executed even though its in the code
/* Multi
 * line
 * comment */
print("This code will actually get executed");
```

1.1.8.3 Importing Functions

In Netscript you can import functions that are declared in other scripts. The script will incur the RAM usage of all imported functions. There are two ways of doing this:

```
import * as namespace from "script filename"; //Import all functions from script
import {fn1, fn2, ...} from "script filename"; //Import specific functions from script
```

Suppose you have a library script called *testlibrary.script*:

```
function fool(args) {
    //function definition...
}

function foo2(args) {
    //function definition...
}

function foo3(args) {
    //function definition...
}

function foo4(args) {
    //function definition...
}
```

Then, if you wanted to use these functions in another script, you can import them like so:

```
import * as testlib from "testlibrary.script";

values = [1,2,3];

//The imported functions must be specified using the namespace
someVal1 = testlib.foo3(values);
someVal2 = testlib.foo1(values);
if (someVal1 > someVal2) {
    //...
} else {
    //...
}
```

If you only wanted to import certain functions, you can do so without needing to specify a namespace for the import:

```
import {foo1, foo3} from "testlibrary.script"; //Saves RAM since not all functions_
↳are imported!

values = [1,2,3];

//No namespace needed
someVal1 = foo3(values);
someVal2 = foo1(values);
if (someVal1 > someVal2) {
    //...
} else {
    //...
}
```

Warning: For those who are experienced with JavaScript, note that the *export* keyword should **NOT** be used in *Netscript 1.0*, as this will break the script. It can, however, be used in *NS2* (but it's not required).

1.1.8.4 Standard, Built-In JavaScript Objects

Standard built-in JavaScript objects such as [Math](#), [Date](#), [Number](#), and others are supported as expected based on which version of Netscript you use (i.e. *Netscript 1.0* will support built-in objects that are defined in ES5, and *NS2* will support whatever your browser supports).

1.2 Basic Gameplay

This section documents Bitburner gameplay elements that are immediately available and/or accessible to the player.

1.2.1 Stats

The player has several stats that can be increased in order to progress in the game.

1.2.1.1 Hacking

Represents the player's ability to code and hack.

Affects:

- Time it takes to hack a server
- Time it takes to execute the grow() and weaken() Netscript function
- Chance to successfully hack a server
- Percent money stolen when hacking a server
- Success rate of certain crimes
- Time it takes to create a program
- Faction reputation gain when carrying out Hacking Contracts or Field Work
- Company reputation gain for certain jobs

Gain experience by:

- Manually hacking servers through Terminal
- Executing hack(), grow(), or weaken() through a script
- Committing certain crimes
- Carrying out Hacking Contracts or doing Field work for Factions
- Working certain jobs at a company
- Studying at a university

1.2.1.2 Strength

Represents the player's physical offensive power

Affects:

- Success rate of certain crimes
- Faction reputation gain for Security and Field Work
- Company reputation gain for certain jobs

Gain experience by:

- Committing certain crimes
- Working out at a gym
- Doing Security/Field Work for a faction
- Working certain jobs at a company

1.2.1.3 Defense

Represents the player's ability to withstand damage

Affects:

- Success rate of certain crimes
- The player's HP
- Faction reputation gain for Security and Field Work
- Company reputation gain for certain jobs

Gain experience by:

- Committing certain crimes
- Working out at a gym
- Doing Security/Field Work for a faction
- Working certain jobs at a company

1.2.1.4 Dexterity

Represents the player's skill and adeptness in performing certain tasks

Affects:

- Success rate of certain crimes
- Faction reputation gain for Security and Field Work
- Company reputation gain for certain jobs

Gain experience by:

- Committing certain crimes
- Working out at a gym
- Doing Security/Field Work for a faction
- Working certain jobs at a company

1.2.1.5 Agility

Represents the player's speed and ability to move

Affects:

- Success rate of certain crimes
- Faction reputation gain for Security and Field Work
- Company reputation gain for certain jobs

Gain experience by:

- Committing certain crimes
- Working out at a gym
- Doing Security/Field Work for a faction
- Working certain jobs at a company

1.2.1.6 Charisma

Represents the player's social abilities

Affects:

- Success rate of certain crimes
- Faction reputation gain for Field Work
- Company reputation gain for most jobs

Gain experience by:

- Committing certain crimes
- Studying at a university
- Working a relevant job at a company
- Doing Field work for a Faction

1.2.2 Terminal

The Terminal is a console emulator program that lets you interface with all of the Servers in the game. The Terminal can be accessed by clicking the ‘Terminal’ tab on the navigation menu on the left-hand side of the game (you may need to expand the ‘Hacking’ header in order to see the ‘Terminal’ tab). Alternatively, the *keyboard shortcut* `Alt + t` can be used to open the Terminal.

1.2.2.1 Filesystem (Directories)

The Terminal contains a **very** basic filesystem that allows you to store and organize your files into different directories. Note that this is **not** a true filesystem implementation. Instead, it is done almost entirely using string manipulation. For this reason, many of the nice & useful features you’d find in a real filesystem do not exist.

Here are the Terminal commands you’ll commonly use when dealing with the filesystem.

- `ls`
- `cd`
- `mv`

1.2.2.1.1 Directories

In order to create a directory, simply name a file using a full absolute Linux-style path:

```
/scripts/myScript.js
```

This will automatically create a “directory” called `scripts`. This will also work for subdirectories:

```
/scripts/hacking/helpers/myHelperScripts.script
```

Files in the root directory do not need to begin with a forward slash:

```
thisIsAFileInTheRootDirectory.txt
```

Note that there is no way to manually create or remove directories. The creation and deletion of directories is automatically handled as you name/rename/delete files.

1.2.2.1.2 Absolute vs Relative Paths

Many Terminal commands accept absolute both absolute and relative paths for specifying a file.

An absolute path specifies the location of the file from the root directory (`/`). Any path that begins with the forward slash is an absolute path:

```
$ nano /scripts/myScript.js
$ cat /serverList.txt
```

A relative path specifies the location of the file relative to the current working directory. Any path that does **not** begin with a forward slash is a relative path. Note that the Linux-style dot symbols will work for relative paths:

```
. (a single dot) - represents the current directory
.. (two dots) - represents the parent directory

$ cd ..
$ nano ../scripts/myScript.js
$ nano ../../helper.js
```

1.2.2.1.3 Netscript

Note that in order to reference a file, *Netscript* functions require the **full** absolute file path. For example

```
run("/scripts/hacking/helpers.myHelperScripts.script");
rm("/logs/myHackingLogs.txt");
rm("thisIsAFileInTheRootDirectory.txt");
```

Note: A full file path **must** begin with a forward slash (/) if that file is not in the root directory.

1.2.2.1.4 Missing Features

These features that are typically in Linux filesystems have not yet been added to the game:

- Tab autocompletion does not work with relative paths
- mv only accepts full filepaths for the destination argument. It does not accept directories

1.2.2.2 Commands

1.2.2.2.1 alias

```
$ alias [-g] [name="value"]
```

Create or display aliases. An alias enables a replacement of a word with another string. It can be used to abbreviate a commonly used command, or commonly used parts of a command. The NAME of an alias defines the word that will be replaced, while the VALUE defines what it will be replaced by. For example, you could create the alias ‘nuke’ for the Terminal command ‘run NUKE.exe’ using the following:

```
$ alias nuke="run NUKE.exe"
```

Then, to run the NUKE.exe program you would just have to enter ‘nuke’ in Terminal rather than the full command. It is important to note that ‘default’ aliases will only be substituted for the first word of a Terminal command. For example, if the following alias was set:

```
$ alias worm="HTTPWorm.exe"
```

and then you tried to run the following terminal command:

```
$ run worm
```

This would fail because the worm alias is not the first word of a Terminal command. To allow an alias to be substituted anywhere in a Terminal command, rather than just the first word, you must set it to be a global alias using the -g flag:

```
$ alias -g worm="HTTPWorm.exe"
```

Now, the ‘worm’ alias will be substituted anytime it shows up as an individual word in a Terminal command.

Entering just the command ‘alias’ without any arguments prints the list of all defined aliases in the reusable form ‘alias NAME=VALUE’ on the Terminal.

The *unalias* Terminal command can be used to remove aliases.

1.2.2.2.2 analyze

Prints details and statistics about the current server. The information that is printed includes basic server details such as the hostname, whether the player has root access, what ports are opened/closed, and also hacking-related information such as an estimated chance to successfully hack, an estimate of how much money is available on the server, etc.

1.2.2.2.3 backdoor

Installs a backdoor on the current server. Root access is required to do this.

Servers will give different bonuses when you install a backdoor. This can pass faction tests or give bonuses such as discounts from companies.

1.2.2.2.4 buy

```
$ buy [-l/program]
```

Purchase a program through the Dark Web. Requires a TOR Router to use.

If this command is ran with the ‘-l’ flag, it will display a list of all programs that can be purchased through the Dark Web, as well as their costs.

Otherwise, the name of the program must be passed in as a parameter. This name is NOT case-sensitive:

```
$ buy brutessh.exe
```

Note that you do not need to be connected to the actual dark web server in order to run this command. You can use this command at any time on the Terminal.

1.2.2.2.5 cat

```
$ cat [filename]
```

Display a message (.msg), literature (.lit), or text (.txt) file:

```
$ cat j1.msg
$ cat foo.lit
$ cat servers.txt
```

1.2.2.2.6 cd

```
$ cd [dir]
```

Change to the specified directory.

See *Filesystem (Directories)* for details on directories.

Note that this command works even for directories that don't exist. If you change to a directory that doesn't exist, it will not be created. A directory is only created once there is a file in it:

```
$ cd scripts/hacking
$ cd /logs
$ cd ..
```

1.2.2.2.7 check

```
$ check [script name] [args...]
```

Print the logs of the script specified by the script name and arguments to the Terminal. Each argument must be separated by a space. **Remember that a running script is uniquely identified both by its name and the arguments that are used to start it.** So, if a script was ran with the following arguments:

```
$ run foo.script 1 2 foodnstuff
```

Then to run the 'check' command on this script you would have to pass the same arguments in:

```
$ check foo.script 1 2 foodnstuff
```

1.2.2.2.8 clear/cls

Clear the Terminal screen, deleting all of the text. Note that this does not delete the user's command history, so using the up and down arrow keys is still valid. Also note that this is permanent and there is no way to undo this. Both 'clear' and 'cls' do the same thing:

```
$ clear
$ cls
```

1.2.2.2.9 connect

```
$ connect [hostname/ip]
```

Connect to a remote server. The hostname of the remote server must be given as the argument to this command. Note that only servers that are immediately adjacent to the current server in the network can be connected to. To see which servers can be connected to, use the 'scan' command.

1.2.2.2.10 download

Downloads a script or text file to your computer (your real-life computer):

```
$ download masterScript.script
$ download importantInfo.txt
```

You can also download all of your scripts/text files as a zip file using the following Terminal commands:

```
$ download *  
$ download *.script  
$ download *.txt
```

1.2.2.2.11 expr

`$ expr [math expression]`

Evaluate a mathematical expression. The expression is evaluated in JavaScript, and therefore all JavaScript operators should be supported.

Examples:

```
$ expr 5.6 * 10 - 123  
$ expr 3 ** 3
```

1.2.2.2.12 free

Display's the memory usage on the current machine. Print the amount of RAM that is available on the current server as well as how much of it is being used.

1.2.2.2.13 hack

Attempt to hack the current server. Requires root access in order to be run.

Related: Hacking Mechanics (TODO Add link here when page gets made)

1.2.2.2.14 help

`$ help [command]`

Display Terminal help information. Without arguments, 'help' prints a list of all valid Terminal commands and a brief description of their functionality. You can also pass the name of a Terminal command as an argument to 'help' to print more detailed information about the Terminal command. Examples:

```
$ help alias  
$ help scan-analyze
```

1.2.2.2.15 home

Connect to your home computer. This will work no matter what server you are currently connected to.

1.2.2.2.16 hostname

Prints the hostname of the server you are currently connected to.

1.2.2.2.17 ifconfig

Prints the IP address of the server you are currently connected to.

1.2.2.2.18 kill

```
$ kill [script name] [args...] $ kill [pid]
```

Kill the script specified by the script filename and arguments OR by its PID.

If you are killing the script using its filename and arguments, then each argument must be separated by a space. Remember that a running script is uniquely identified by both its name and the arguments that are used to start it. So, if a script was ran with the following arguments:

```
$ run foo.script 50e3 sigma-cosmetics
```

Then to kill this script the same arguments would have to be used:

```
$ kill foo.script 50e3 sigma-cosmetics
```

If you are killing the script using its PID, then the PID argument must be numeric.

1.2.2.2.19 killall

Kills all scripts on the current server.

1.2.2.2.20 ls

```
$ ls [dir] [--grep pattern]
```

Prints files and directories on the current server to the Terminal screen.

If this command is run with no arguments, then it prints all files and directories on the current server to the Terminal screen. Directories will be printed first in alphabetical order, followed by the files (also in alphabetical order).

The `dir` optional parameter allows you to specify the directory for which to display files.

The `--grep pattern` optional parameter allows you to only display files and directories with a certain pattern in their names.

The `-l` optional parameter allows you to force each item onto a single line.

Examples:

```
// List files/directories with the '.script' extension in the current directory
$ ls -l --grep .script

// List files/directories with the '.js' extension in the root directory
$ ls / -l --grep .js

// List files/directories with the word 'purchase' in the name, in the
↳:code:`scripts` directory
$ ls scripts -l --grep purchase
```

1.2.2.2.21 lscpu

Prints the number of CPU cores the current server has.

1.2.2.2.22 mem

`$ mem [script name] [-t] [num threads]`

Displays the amount of RAM needed to run the specified script with a single thread. The command can also be used to print the amount of RAM needed to run a script with multiple threads using the `-t` flag. If the `-t` flag is specified, then an argument for the number of threads must be passed in afterwards. Examples:

```
$ mem foo.script
$ mem foo.script -t 50
```

The first example above will print the amount of RAM needed to run `foo.script` with a single thread. The second example above will print the amount of RAM needed to run `foo.script` with 50 threads.

1.2.2.2.23 mv

`$ mv [source] [destination]`

Move the source file to the specified destination in the filesystem. See [Filesystem \(Directories\)](#) for more details about the Terminal's filesystem. This command only works for scripts and text files (.txt). It cannot, however, be used to convert from script to text file, or vice versa.

This function can also be used to rename files.

Note: Unlike the Linux `mv` command, the *destination* argument must be the full filepath. It cannot be a directory.

Examples:

```
$ mv hacking.script scripts/hacking.script
$ mv myScript.js myOldScript.js
```

1.2.2.2.24 nano

`$ nano [filename]`

Opens up the specified file in the Text Editor. Only scripts (.script, .js) and text files (.txt) can be edited. If the file does not already exist, then a new empty file will be created.

1.2.2.2.25 ps

`$ ps [-g, -grep pattern]`

Prints all scripts that are currently running on the current server. The `-g`, `--grep pattern` option will only output running scripts where the name matches the provided pattern.

1.2.2.26 rm

```
$ rm [filename]
```

Removes the specified file from the current server. This works for every file type except literature files (.lit).

WARNING: This is permanent and cannot be undone

1.2.2.27 run

```
$ run [file name] [-t] [num threads] [args...]
```

Execute a program, script, or *Coding Contracts*.

The ‘[-t]’, ‘[num threads]’, and ‘[args...]’ arguments are only valid when running a script. The ‘-t’ flag is used to indicate that the script should be run with the specified number of threads. If the flag is omitted, then the script will be run with a single thread by default. If the ‘-t’ flag is used, then it **MUST** come immediately after the script name, and the [num threads] argument **MUST** come immediately afterwards.

[args...] represents a variable number of arguments that will be passed into the script. See the documentation about script arguments. Each specified argument must be separated by a space.

Examples

Run a program:

```
$ run BruteSSH.exe
```

Run *foo.script* with 50 threads and the arguments [1e3, 0.5, foodnstuff]:

```
$ run foo.script -t 50 1e3 0.5 foodnstuff
```

Run a Coding Contract:

```
$ run foo-contract.cct
```

1.2.2.28 scan

Prints all immediately-available network connections. This will print a list of all servers that you can currently connect to using the ‘connect’ Terminal command.

1.2.2.29 scan-analyze

```
$ scan-analyze [depth]
```

Prints detailed information about all servers up to *[depth]* nodes away on the network. Calling ‘scan-analyze 1’ will display information for the same servers that are shown by the ‘scan’ Terminal command. This command also shows the relative paths to reach each server.

By default, the maximum depth that can be specified for ‘scan-analyze’ is 3. However, once you have the *DeepscanV1.exe* and *DeepscanV2.exe* programs, you can execute ‘scan-analyze’ with a depth up to 5 and 10, respectively.

The information ‘scan-analyze’ displays about each server includes whether or not you have root access to it, its required hacking level, the number of open ports required to run NUKE.exe on it, and how much RAM it has.

1.2.2.2.30 scp

```
$ scp [script name] [target server]
```

Copies the specified script from the current server to the target server. The second argument passed in must be the hostname or IP of the target server.

1.2.2.2.31 sudov

Prints whether or not you have root access to the current server.

1.2.2.2.32 tail

```
$ tail [script name] [args...]
```

Displays dynamic logs for the script specified by the script name and arguments. Each argument must be separated by a space. Remember that a running script is uniquely identified by both its name and the arguments that were used to run it. So, if a script was ran with the following arguments:

```
$ run foo.script 10 50000
```

Then in order to check its logs with ‘tail’ the same arguments must be used:

```
$ tail foo.script 10 50000
```

1.2.2.2.33 top

Prints a list of all scripts running on the current server as well as their thread count and how much RAM they are using in total.

1.2.2.2.34 unalias

```
$ unalias "[alias name]"
```

Deletes the specified alias. Note that the double quotation marks are required.

As an example, if an alias was declared using:

```
$ alias r="run"
```

Then it could be removed using:

```
$ unalias "r"
```

It is not necessary to differentiate between global and non-global aliases when using ‘unalias’

1.2.2.2.35 wget

```
$ wget [url] [target file]
```

Retrieves data from a url and downloads it to a file on the current server. The data can only be downloaded to a script (.script, .js) or a text file (.txt). If the target file already exists, it will be overwritten by this command.

Note that will not be possible to download data from many websites because they do not allow cross-origin resource sharing (CORS). This includes websites such as gist and pastebin. One notable site it will work on is rawgithub. Example:

```
$ wget https://raw.githubusercontent.com/danielyxie/bitburner/master/README.md game_
↪readme.txt
```

1.2.2.3 Argument Parsing

When evaluating a terminal command, arguments are initially parsed based on whitespace (usually spaces). Each whitespace character signifies the end of an argument, and potentially the start of new one. For most terminal commands, this is all you need to know.

When running scripts, however, it is important to know in more detail how arguments are parsed. There are two main points:

1. Quotation marks can be used to wrap a single argument and force it to be parsed as a string. Any whitespace inside the quotation marks will not cause a new argument to be parsed.
2. Anything that can represent a number is automatically cast to a number, unless its surrounded by quotation marks.

Here's an example to show how these rules work. Consider the following script *argType.script*:

```
tprint("Number of args: " + args.length);
for (var i = 0; i < args.length; ++i) {
  tprint(typeof args[i]);
}
```

Then if we run the following terminal command:

```
$ run argType.script 123 1e3 "5" "this is a single argument"
```

We'll see the following in the Terminal:

```
Running script with 1 thread(s) and args: [123, 1000, "5", "this is a single argument
↪"].
May take a few seconds to start up the process...
argType.script: Number of args: 4
argType.script: number
argType.script: number
argType.script: string
argType.script: string
```

1.2.2.4 Chaining Commands

You can run multiple Terminal commands at once by separating each command with a semicolon (;).

Example:

```
$ run foo.script; tail foo.script
```

1.2.3 Servers

In this game, a server refers to a computer that can be connected to, accessed, and manipulated through the Terminal. All servers in the game are connected to each other to form a large, global network. To learn about how to navigate this network and connect to other servers, see the [Terminal](#) page.

1.2.3.1 Server RAM

Perhaps the most important property of a server to make note of is its RAM, which refers to how much memory is available on that machine. RAM is important because it is required to run Scripts. More RAM allows the user to run more powerful and complicated scripts as well as executing a script with [more threads](#).

The *free*, *scan-analyze*, and *analyze* Terminal commands can be used to check how much RAM a server has.

1.2.3.2 Identifying Servers

A server is identified by its hostname. A hostname is a label assigned to a server. A hostname will usually give you a general idea of what the server is. For example, the company Nova Medical might have a server with the hostname “nova-med”.

Hostnames are unique. This means that if one server has the the hostname “some-server”, then no other server in the game can have that that hostname.

There are many [Netscript Functions](#) and [Terminal](#) commands in the game that will require you to target a specific server by hostname.

1.2.3.3 Player-owned Servers

The player starts with a single server: his/her home computer. This server will have the hostname “home.” The player’s home computer is special for a variety of reasons:

1. The home computer’s RAM can be upgraded. This can be done by visiting certain locations in the World.
2. The home computer persists through Augmentation Installations. This means that you will not lose any RAM upgrades or Scripts on your home computer when you install [Augmentations](#) (you will however, lose programs and messages on your home computer).

The player can also purchase additional servers. This can be done by visiting certain locations in the World, or it can be done automatically through a script using the [purchaseServer\(\)](#) Netscript Function. The advantage of purchased servers is that, in terms of RAM, they are cheaper than upgrading your home computer. The disadvantage is that your purchased servers are lost when you install Augmentations.

1.2.3.4 Hackable Servers

Most servers that are not owned by the player can be hacked for money and exp. See the [Hacking](#) page for more details.

Different servers have different levels of security, but also offer different rewards when being hacked.

1.2.4 Hacking

In the year 2077, currency has become digital and decentralized. People and corporations store their money on servers. By hacking these servers, you can steal their money and gain experience.

1.2.4.1 Gaining Root Access

The first step to hacking a server is to gain root access to that server. This can be done using the NUKE virus (NUKE.exe). You start the game with a copy of the NUKE virus on your home computer. The NUKE virus attacks the target server's open ports using buffer overflow exploits. When successful, you are granted root administrative access to the machine.

In order for the NUKE virus to succeed, the target server needs to have enough open ports. Some servers have no security and will not need any ports opened. Some will have very high security and will need many ports opened. In order to open ports on another server, you will need to run programs that attack the server to open specific ports. These programs can be coded once your hacking skill gets high enough, or they can be purchased if you can find a seller.

There are two ways to execute port-opening programs and the NUKE virus:

1. Connect to the target server through the *Terminal* and use the *run* Terminal command:

```
$ run [programName]
```

2. Use a *Netscript Function*:

- *nuke()*
- *brutessh()*
- *ftpcrack()*
- *relaysmtp()*
- *httpworm()*
- *sqlinject()*

There are two ways to determine how many ports need to be opened on a server in order to successfully NUKE it:

1. Connect to that server through the *Terminal* and use the *analyze* command
2. Use the *getServerNumPortsRequired()* Netscript function

Once you have enough ports opened on a server and have ran the NUKE virus to gain root access, you will be able to hack it.

1.2.4.2 General Hacking Mechanics

When you execute the hack command, either manually through the terminal or automatically through a script, you attempt to hack the server. This action takes time. The more advanced a server's security is, the more time it will take. Your hacking skill level also affects the hacking time, with a higher hacking skill leading to shorter hacking times. Also, running the hack command manually through terminal is faster than hacking from a script.

Your attempt to hack a server will not always succeed. The chance you have to successfully hack a server is also determined by the server's security and your hacking skill level. Even if your hacking attempt is unsuccessful, you will still gain experience points.

When you successfully hack a server. You steal a certain percentage of that server's total money. This percentage is, once again, determined by the server's security and your hacking skill level. The amount of money on a server is not limitless. So, if you constantly hack a server and deplete its money, then you will encounter diminishing returns in your hacking (since you are only hacking a certain percentage). You can increase the amount of money on a server using a script and the *grow()* function in Netscript.

1.2.4.3 Server Security

Each server has a security level, typically between 1 and 100. A higher number means the server has stronger security. It is possible for a server to have a security of level 100 or higher, in which case hacking that server will become impossible (0% chance for hack to succeed).

As mentioned above, a server's security level is an important factor to consider when hacking. You can check a server's security level using the *analyze* Terminal command. You can also check a server's security in a script, using the *getServerSecurityLevel()* Netscript Function. See the Netscript documentation for more details.

Whenever a server is hacked manually or through a script, its security level increases by a small amount. Calling the *grow()* function in a script will also increase security level of the target server. These actions will make it harder for you to hack the server, and decrease the amount of money you can steal. You can lower a server's security level in a script using the *weaken()* function in Netscript. See the Netscript documentation for more details

A server has a minimum security level that is equal to one third of its starting security, rounded to the nearest integer. To be more precise:

```
server.minSecurityLevel = Math.max(1, Math.round(server.startingSecurityLevel / 3))
```

This means that a server's security level will not fall below this value if you are trying to weaken() it.

1.2.4.4 Backdoors

Servers that can be hacked can also have backdoors installed. These backdoors will provide you with a benefit; the services may be cheaper, penalties may be reduced or there may be other results. Honeypots exist and will let factions know when you have succeeded at backdooring their system. Once you have a backdoor installed, you can connect to that server directly.

When you visit a location in the city and see that the name is partially scrambled, this indicates that you have backdoored the server related to the location.

1.2.5 Scripts

Scripts are programs that can be used to automate the hacking process and almost every other part of the game. Scripts must be written in the *Netscript* language.

It is highly recommended that you have a basic background in programming to start writing scripts. You by no means need to be an expert. All you need is some familiarity with basic programming constructs like for/while loops, conditionals (if/else), functions, variables, etc. If you'd like to learn a little bit about programming, see *Learn to Program in Netscript*.

1.2.5.1 Script Arguments

When running a script, you can choose to pass arguments to that script. The script's logic can access and act on these arguments. This allows for flexibility in your scripts. For more details, see *Netscript Script Arguments*.

For information on how to run scripts with arguments, see *Working with Scripts in Terminal* and *Working with Scripts in Netscript* below.

1.2.5.2 Identifying a Script

Many commands and functions act on an executing script (i.e. a script that is running). Therefore, there must be a way to specify which script you want those commands & functions to act on.

A script that is being executed is uniquely identified by both its name and the arguments that it was run with.

The arguments must be an **exact** match. This means that both the order and type of the arguments matter.

1.2.5.3 Multithreading scripts

A script can be run with multiple threads. This is also called multithreading. The effect of multithreading is that every call to the *hack()*, *grow()*, and *weaken()* Netscript functions will have their results multiplied by the number of threads. For example, if a normal single-threaded script is able to hack \$10,000, then running the same script with 5 threads would yield \$50,000.

(This is the **only** affect of running a script with multiple threads. Scripts will not actually become multithreaded in the real-world sense.)

When multithreading a script, the total RAM cost can be calculated by simply multiplying the base RAM cost of the script with the number of threads, where the base cost refers to the amount of RAM required to run the script single-threaded. In the terminal, you can run the *mem* Terminal command to see how much RAM a script requires with *n* threads:

```
$ mem [scriptname] -t n
```

1.2.5.4 Working with Scripts in Terminal

Running a script requires RAM. The more complex a script is, the more RAM it requires to run. Scripts can be run on any server you have root access to.

Here are some *Terminal* commands that are useful when working with scripts:

check [script] [args...]

Prints the logs of the script specified by the name and arguments to Terminal. Arguments should be separated by a space. Remember that scripts are uniquely identified by their arguments as well as their name. For example, if you ran a script *foo.script* with the argument *foodnstuff* then in order to ‘check’ it you must also add the *foodnstuff* argument to the check command:

```
$ check foo.script foodnstuff
```

free

Shows the current server’s RAM usage and availability

kill [script] [args...]

Stops a script that is running with the specified script name and arguments. Arguments should be separated by a space. Remember that scripts are uniquely identified by their arguments as well as their name. For example, if you ran a script *foo.script* with the argument 1 and 2, then just typing “*kill foo.script*” will not work. You have to use:

```
$ kill foo.script 1 2
```

mem [script] [-t] [n]

Check how much RAM a script requires to run with *n* threads

nano [script]

Create/Edit a script. The name of the script must end with a valid extension: *.script*, or *.js*

ps

Displays all scripts that are actively running on the current server

rm [script]

Delete a script from the server. This is permanent

run [script] [-t] [n] [args...]

Run a script with n threads and the specified arguments. Each argument should be separated by a space. Both the arguments and thread specification are optional. If neither are specified, then the script will be run single-threaded with no arguments.

Examples:

Run 'foo.script' single-threaded with no arguments:

```
$ run foo.script
```

Run 'foo.script' with 10 threads and no arguments:

```
$ run foo.script -t 10
```

Run 'foo.script' single-threaded with three arguments: [foodnstuff, sigma-cosmetics, 10]:

```
$ run foo.script foodnstuff sigma-cosmetics 10
```

Run 'foo.script' with 50 threads and a single argument: [foodnstuff]:

```
$ run foo.script -t 50 foodnstuff
```

tail [script] [args...]

Displays the logs of the script specified by the name and arguments. Note that scripts are uniquely identified by their arguments as well as their name. For example, if you ran a script 'foo.script' with the argument 'foodnstuff' then in order to 'tail' it you must also add the 'foodnstuff' argument to the tail command as so: tail foo.script foodnstuff

top

Displays all active scripts and their RAM usage

1.2.5.5 Working with Scripts in Netscript

TODO/Coming Soon...

1.2.5.6 Notes about how Scripts Work Offline

The scripts that you write and execute are interpreted in Javascript. For this reason, it is not possible for these scripts to run while offline (when the game is closed). It is important to note that for this reason, conditionals such as if/else statements and certain commands such as purchaseHacknetNode() or nuke() will not work while the game is offline.

However, Scripts WILL continue to generate money and hacking exp for you while the game is offline. This offline production is based off of the scripts' production while the game is online.

grow() and weaken() are two Netscript commands that will also be applied when the game is offline, although at a slower rate compared to if the game was open. This is done by having each script keep track of the rate at which the grow() and weaken() commands are called when the game is online. These calculated rates are used to determine how many times these function calls would be made while the game is offline.

Also, note that because of the way the Netscript interpreter is implemented, whenever you reload or re-open the game all of the scripts that you are running will start running from the BEGINNING of the code. The game does not keep track of where exactly the execution of a script is when it saves/loads.

1.2.6 World

In Bitburner, the world consists of six different cities:

- Sector-12 (this is where you start out)
- Aevum
- Ishima
- New Tokyo
- Chongqing
- Volhaven

1.2.7 Factions

Warning: This page contains spoilers regarding the game's story/plot-line.

Throughout the game you may receive invitations from factions. There are many different factions, and each faction has different criteria for determining its potential members. Joining a faction and furthering its cause is crucial to progressing in the game and unlocking endgame content.

It is possible to join multiple factions if you receive invitations from them. However, note that joining a faction may prevent you from joining other rival factions. (Don't worry, this usually isn't the case. Also, it would only be temporary since resetting the game by installing *Augmentations* will clear all your factions)

The 'Factions' link on the menu brings up a list of all factions that you have joined. You can select a Faction on this list to go to that Faction page. This page displays general information about the Faction and also lets you perform work for the faction. Working for a Faction is similar to working for a company except that you don't get paid a salary. You will only earn reputation in your Faction and train your stats. Also, cancelling work early when working for a Faction does **not** result in reduced reputation earnings.

Earning reputation for a Faction unlocks powerful Augmentations. Purchasing and installing these Augmentations will upgrade your abilities. The Augmentations that are available to unlock vary from faction to faction.

1.2.7.1 List of Factions and their Requirements

Early Game Factions	Faction Name	Requirements	Joining this Faction prevents you from joining:
	CyberSec	<ul style="list-style-type: none"> • Install a backdoor on the CSEC server 	
	Tian Di Hui	<ul style="list-style-type: none"> • \$1m • Hacking Level 50 • Be in Chongqing, New Tokyo, or Ishima 	
	Netburners	<ul style="list-style-type: none"> • Hacking Level 80 • Total Hacknet Levels of 100 • Total Hacknet RAM of 8 • Total Hacknet Cores of 4 	

City Factions	Faction Name	Requirements	Joining this Faction prevents you from joining:
	Sector-12	<ul style="list-style-type: none"> • Be in Sector-12 • \$15m 	<ul style="list-style-type: none"> • Chongqing • New Tokyo • Ishima • Volhaven
	Chongqing	<ul style="list-style-type: none"> • Be in Chongqing • \$20m 	<ul style="list-style-type: none"> • Sector-12 • Aevum • Volhaven
	New Tokyo	<ul style="list-style-type: none"> • Be in New Tokyo • \$20m 	<ul style="list-style-type: none"> • Sector-12 • Aevum • Volhaven
	Ishima	<ul style="list-style-type: none"> • Be in Ishima • \$30m 	<ul style="list-style-type: none"> • Sector-12 • Aevum • Volhaven
	Aevum	<ul style="list-style-type: none"> • Be in Aevum • \$40m 	<ul style="list-style-type: none"> • Chongqing • New Tokyo • Ishima • Volhaven
	Volhaven	<ul style="list-style-type: none"> • Be in Volhaven • \$50m 	<ul style="list-style-type: none"> • Sector-12 • Aevum • Chongqing • New Tokyo • Ishima

Hacking Groups	Faction Name	Requirements	Joining this Faction prevents you from joining:
	NiteSec	<ul style="list-style-type: none"> • Install a backdoor on the avmnite-02h server 	
	The Black Hand	<ul style="list-style-type: none"> • Install a backdoor on the I.I.I.I server 	
	BitRunners	<ul style="list-style-type: none"> • Install a backdoor on the run4teh111z server 	

Megacorporations	Faction Name	Requirements	Joining this Faction prevents you from joining:
	ECorp	<ul style="list-style-type: none"> Have 200k reputation with the Corporation 	
	MegaCorp	<ul style="list-style-type: none"> Have 200k reputation with the Corporation 	
	KuaiGong International	<ul style="list-style-type: none"> Have 200k reputation with the Corporation 	
	Four Sigma	<ul style="list-style-type: none"> Have 200k reputation with the Corporation 	
	NWO	<ul style="list-style-type: none"> Have 200k reputation with the Corporation 	
	Blade Industries	<ul style="list-style-type: none"> Have 200k reputation with the Corporation 	
	OmniTek Incorporated	<ul style="list-style-type: none"> Have 200k reputation with the Corporation 	
	Bachman & Associates	<ul style="list-style-type: none"> Have 200k reputation with the Corporation 	
	Clarke Incorporated	<ul style="list-style-type: none"> Have 200k reputation with the Corporation 	
	Fulcrum Secret Technologies	<ul style="list-style-type: none"> Have 250k reputation with the Corporation Install a backdoor on the fulcrumasets server 	

Criminal Organizations	Faction Name	Requirements	Joining this Faction prevents you from joining:
	Slum Snakes	<ul style="list-style-type: none"> • All Combat Stats of 30 • -9 Karma • \$1m 	
	Tetrads	<ul style="list-style-type: none"> • Be in Chongqing, New Tokyo, or Ishima • All Combat Stats of 75 • -18 Karma 	
	Silhouette	<ul style="list-style-type: none"> • CTO, CFO, or CEO of a company • \$15m • -22 Karma 	
	Speakers for the Dead	<ul style="list-style-type: none"> • Hacking Level 100 • All Combat Stats of 300 • 30 People Killed • -45 Karma • Not working for CIA or NSA 	
	The Dark Army	<ul style="list-style-type: none"> • Hacking Level 300 • All Combat Stats of 300 • Be in Chongqing • 5 People Killed • -45 Karma • Not working for CIA or NSA 	
	The Syndicate	<ul style="list-style-type: none"> • Hacking Level 200 • All Combat Stats of 200 • Be in Aevum or Sector-12 • \$10m • -90 Karma • Not working for CIA or NSA 	

Endgame Factions	Faction Name	Requirements	Joining this Faction prevents you from joining:
	The Covenant	<ul style="list-style-type: none"> • 20 Augmentations • \$75b • Hacking Level of 850 • All Combat Stats of 850 	
	Daedalus	<ul style="list-style-type: none"> • 30 Augmentations • \$100b • Hacking Level of 2500 OR All Combat Stats of 1500 	
	Illuminati	<ul style="list-style-type: none"> • 30 Augmentations • \$150b • Hacking Level of 1500 • All Combat Stats of 1200 	

1.2.8 Augmentations

Advances in science and medicine have led to powerful new technologies that allow people to augment themselves beyond normal human capabilities. There are many different types of Augmentations, ranging from cybernetic to genetic to biological. Acquiring these Augmentations enhances the user’s physical and mental faculties.

Augmentations provide persistent upgrades in the form of multipliers. These multipliers apply to a wide variety of things such as stats, experience gain, and hacking, just to name a few. Your multipliers can be viewed in the ‘Character’ page (*keyboard shortcut* Alt + c).

1.2.8.1 How to acquire Augmentations

Because of how powerful Augmentations are, the technology behind them is kept private and secret by the corporations and organizations that create them. Therefore, the only way for the player to obtain Augmentations is through Factions. After joining a Faction and earning enough reputation in it, you will be able to purchase its Augmentations. Different Factions offer different Augmentations. Augmentations must be purchased in order to be installed, and they are fairly expensive.

1.2.8.2 Installing Augmentations

You will not gain the benefits of your purchased Augmentations until you actually install them. You can choose to install Augmentations through the ‘Augmentations’ menu tab (Found under ‘Character’. Alternatively, use the keyboard shortcut Alt + a).

Unfortunately, installing Augmentations has side effects. You will lose most of the progress you’ve made, including your skills, stats, and money. You will have to start over, but you will have all of the Augmentations you have installed to help you progress. This is the game’s “soft reset” or “prestige” mechanic.

To summarize, here is a list of everything you will LOSE when you install an Augmentation:

- Stats/Skills
- Money
- Scripts on all servers EXCEPT your home computer
- Purchased servers
- Hacknet Nodes
- Company/faction reputation
- Jobs and Faction memberships
- Programs
- Stocks
- TOR router

Here is everything you will KEEP when you install an Augmentation:

- Every Augmentation you have installed
- Scripts on your home computer
- RAM/Core Upgrades on your home computer
- World Stock Exchange account and TIX API Access

1.2.8.3 Purchasing Multiple Augmentations

You do not have to install an Augmentation right after you purchase it. You can purchase as many Augmentations as you'd like before you choose to install them. When you install your purchased Augmentations they will ALL get installed.

There are a few drawbacks to this, however. First, obviously, you won't gain the benefits of your purchased Augmentations until after you install them. Second, purchasing multiple Augmentations before installing them will cause the Augmentations to get progressively more expensive. When you purchase an Augmentation, the price of purchasing another Augmentation doubles. This multiplier stacks for each Augmentation you purchase. Once you install your purchased Augmentations, their costs are reset back to the original prices.

1.2.9 Companies

When exploring the [world](#), you can visit various companies. At these companies, you can apply for jobs.

Working a job lets you earn money, experience, and reputation with that company.

While working for a company, you can click "Do something else simultaneously" to be able to do things while you continue to work in the background. There is a 20% penalty to the related gains. Clicking the "Focus" button under the overview will return you to the current work.

Reputation is required to apply for a promotion. This reputation is not counted towards your career until the shift ends, either due to the time spent or clicking the "Stop Working" button. For most jobs there is a penalty of 50% of the reputation gained if you stop your shift early.

1.2.9.1 Information about all Companies

TODO

1.2.10 Crimes

Committing crimes is an active gameplay mechanic that allows the player to train their stats and potentially earn money. The player can attempt to commit crimes by visiting ‘The Slums’ through the ‘City’ tab (*Keyboard shortcut* Alt + w). ‘The Slums’ is available in every city.

1.2.10.1 Basic Mechanics

When you visit the ‘Slums’ you will see a list of buttons that show all of the available crimes. Simply select one of the options to begin attempting that crime. Attempting to commit a crime takes a certain amount of time. This time varies between crimes. During this time, you cannot do anything else.

Crimes are not always successful. Your rate of success is determined by your stats (and Augmentation multipliers) and can be seen on the crime-selection page. If you are unsuccessful at committing a crime you will gain EXP, but you will not earn money. If you are successful at committing the crime you will gain extra EXP (double of what an unsuccessful attempt would give) and earn money.

Harder crimes are typically more profitable, and also give more EXP.

1.2.10.2 Crime details

Available crimes, and their descriptions, which all begin with “attempt to...” Shoplift ...shoplift from a low-end retailer Rob store ...commit armed robbery on a high-end store Mug someone ...mug a random person on the street Larceny ...rob property from someone’s house Deal Drugs ...deal drugs Bond Forgery ...forge corporate bonds Traffick illegal Arms ...smuggle illegal arms into the city Homicide ...murder a random person on the street Grand theft Auto ...commit grand theft auto Kidnap and Ransom ...kidnap and ransom a high-profile-target Assassinate ...assassinate a high-profile target Heist ...pull off the ultimate heist

1.2.11 Infiltration

Infiltration is a gameplay mechanic that allows you to infiltrate a company’s facility to try and steal the company’s classified secrets. These secrets can be sold for money or for reputation with a faction.

1.2.11.1 Overview

Many companies have facilities that you can attempt to infiltrate. By infiltrating, you can steal classified company secrets and then sell these for money or for faction reputation. To try and infiltrate a company, visit a company through the ‘World’ menu. There will be an option that says ‘Infiltrate Company’.

When infiltrating a company you will be presented with short active challenges. None of the challenges use the mouse.

The difficulty at the top lowers with better combat stats and charisma. It is not recommended to attempt infiltrations above mid-normal.

The “maximum level” is the number of challenges you will need to pass to receive the infiltration reward.

Every time you fail an infiltration challenge, you will take damage based on the difficulty of the infiltration. If you are reduced to 0 hp or below, the infiltration will immediately end.

- Most use spacebar as “action”
- Some use WASD or arrows interchangeably.
- A few others use the rest of the keyboard.

**** Slash when his guard is down! ****

Press space when the guard is attacking you. There's 3 phase The first is guarding, where attacking back will result in failure. The 2nd is preparing, this informs you that in 250ms there will be an opening window to attack. The 3rd is attack, during this phase you can press space to slash and kill the enemy.

**** Close the brackets ****

Enter all the matching brackets in reverse order.

**** Type it backward ****

Type the words that are written backward.

**** Say something nice about the guard. ****

Use the arrows to find a compliment for the guard.

**** Enter the Code! ****

Match the arrows as they appears.

**** Match the symbols! ****

Move the cursor to the matching symbol and press space to confirm.

**** Remember all the mines! ****

At first the cursor cannot be moved, remember the positions of the X. Then move the cursor and press space to mark the mines on the board.

**** Cut the wires ****

Follow the instructions and press the numbers 1 through 9 to cut the appropriate wires.

1.2.12 Stock Market

The Stock Market refers to the World Stock Exchange (WSE), through which you can buy and sell stocks in order to make money.

The WSE can be found in the 'City' tab, and is accessible in every city.

1.2.12.1 Fundamentals

The Stock Market is not as simple as "buy at price X and sell at price Y". The following are several fundamental concepts you need to understand about the stock market.

Note: For those that have experience with finance/trading/investing, please be aware that the game's stock market does not function exactly like it does in the real world. So these concepts below should seem similar, but won't be exactly the same.

1.2.12.1.1 Positions: Long vs Short

When making a transaction on the stock market, there are two types of positions: Long and Short. A Long position is the typical scenario where you buy a stock and earn a profit if the price of that stock increases. Meanwhile, a Short position is the exact opposite. In a Short position you purchase shares of a stock and earn a profit if the price of that stock decreases. This is also called 'shorting' a stock.

Note: Shorting stocks is not available immediately, and must be unlocked later in the game.

1.2.12.1.2 Forecast & Second-Order Forecast

A stock's forecast is its likelihood of increasing or decreasing in value. The forecast is typically represented by its probability of increasing in either a decimal or percentage form. For example, a forecast of 70% means the stock has a 70% chance of increasing and a 30% chance of decreasing.

A stock's second-order forecast is the target value that its forecast trends towards. For example, if a stock has a forecast of 60% and a second-order forecast of 70%, then the stock's forecast should slowly trend towards 70% over time. However, this is determined by RNG so there is a chance that it may never reach 70%.

Both the forecast and the second-order forecast change over time.

A stock's forecast can be viewed after purchasing Four Sigma (4S) Market Data access. This lets you see the forecast info on the Stock Market UI. If you also purchase access to the 4S Market Data TIX API, then you can view a stock's forecast using the `getStockForecast()` function.

A stock's second-order forecast is always hidden.

1.2.12.1.3 Spread (Bid Price & Ask Price)

The **bid price** is the maximum price at which someone will buy a stock on the stock market.

The **ask price** is the minimum price that a seller is willing to receive for a stock on the stock market

The ask price will always be higher than the bid price (This is because if a seller is willing to receive less than the bid price, that transaction is guaranteed to happen). The difference between the bid and ask price is known as the **spread**. A stock's "price" will be the average of the bid and ask price.

The bid and ask price are important because these are the prices at which a transaction actually occurs. If you purchase a stock in the long position, the cost of your purchase depends on that stock's ask price. If you then try to sell that stock (still in the long position), the price at which you sell is the stock's bid price. Note that this is reversed for a short position. Purchasing a stock in the short position will occur at the stock's bid price, and selling a stock in the short position will occur at the stock's ask price.

1.2.12.1.4 Transactions Influencing Stock Forecast

Buying or selling a large number of shares of a stock will influence that stock's forecast & second-order forecast. The forecast is the likelihood that the stock will increase or decrease in price. The magnitude of this effect depends on the number of shares being transacted. More shares will have a bigger effect.

The effect that transactions have on a stock's second-order forecast is significantly smaller than the effect on its forecast.

1.2.12.1.5 Order Types

There are three different types of orders you can make to buy or sell stocks on the exchange: Market Order, Limit Order, and Stop Order.

Note: Limit Orders and Stop Orders are not available immediately, and must be unlocked later in the game.

When you place a Market Order to buy or sell a stock, the order executes immediately at whatever the current price of the stock is. For example if you choose to short a stock with 5000 shares using a Market Order, you immediately purchase those 5000 shares in a Short position at whatever the current market price is for that stock.

A Limit Order is an order that only executes under certain conditions. A Limit Order is used to buy or sell a stock at a specified price or better. For example, lets say you purchased a Long position of 100 shares of some stock at a price of \$10 per share. You can place a Limit Order to sell those 100 shares at \$50 or better. The Limit Order will execute when the price of the stock reaches a value of \$50 or higher.

A Stop Order is the opposite of a Limit Order. It is used to buy or sell a stock at a specified price (before the price gets 'worse'). For example, lets say you purchased a Short position of 100 shares of some stock at a price of \$100 per share. The current price of the stock is \$80 (a profit of \$20 per share). You can place a Stop Order to sell the Short position if the stock's price reaches \$90 or higher. This can be used to lock in your profits and limit any losses.

Here is a summary of how each order works and when they execute:

In a LONG Position:

A Limit Order to buy will execute if the stock's price \leq order's price

A Limit Order to sell will execute if the stock's price \geq order's price

A Stop Order to buy will execute if the stock's price \geq order's price

A Stop Order to sell will execute if the stock's price \leq order's price

In a SHORT Position:

A Limit Order to buy will execute if the stock's price \geq order's price

A Limit Order to sell will execute if the stock's price \leq order's price

A Stop Order to buy will execute if the stock's price \leq order's price

A Stop Order to sell will execute if the stock's price \geq order's price.

1.2.12.1.6 Player Actions Influencing Stocks

It is possible for your actions elsewhere in the game to influence the stock market.

Hacking If a server has a corresponding stock (e.g. *foodnstuff* server -> FoodNStuff stock), then hacking that server can decrease the stock's second-order forecast. This causes the corresponding stock's forecast to trend downwards in value over time.

This effect only occurs if you set the *stock* option to true when calling the *hack()* function. The chance that hacking a server will cause this effect is based on what percentage of the server's total money you steal.

A single hack will have a minor effect, but continuously hacking a server for lots of money over time will have a noticeable effect in making the stock's forecast trend downwards.

Growing If a server has a corresponding stock (e.g. *foodnstuff* server -> FoodNStuff stock), then growing that server's money can increase the stock's second-order forecast. This causes the corresponding stock's forecast to trend upwards in value over time.

This effect only occurs if you set the *stock* option to true when calling the *grow()* function. The chance that growing a server will cause this effect is based on what percentage of the server's total money to add to it.

A single grow operation will have a minor effect, but continuously growing a server for lots of money over time will have a noticeable effect in making the stock's forecast trend upwards.

Working for a Company If a company has a corresponding stock, then working for that company will increase the corresponding stock's second-order forecast. This will cause the stock's forecast to (slowly) trend upwards in value over time.

The potency of this effect is based on how “effective” you are when you work (i.e. its based on your stats and multipliers).

1.2.12.2 Automating the Stock Market

You can write scripts to perform automatic and algorithmic trading on the Stock Market. See [TIX API](#) for more details.

1.2.12.3 Under the Hood

Stock prices are updated every ~6 seconds.

Whether a stock’s price moves up or down is determined by RNG. However, stocks have properties that can influence the way their price moves. These properties are hidden, although some of them can be made visible by purchasing the Four Sigma (4S) Market Data upgrade. Some examples of these properties are:

- Volatility
- Likelihood of increasing or decreasing (i.e. the stock’s forecast)
- Likelihood of forecast increasing or decreasing (i.e. the stock’s second-order forecast)
- How easily a stock’s price/forecast is influenced by transactions
- Spread percentage
- Maximum price (not a real maximum, more of a “soft cap”)

Each stock has its own unique values for these properties.

1.2.12.4 Offline Progression

The Stock Market does not change or process anything while the game has closed. However, it does accumulate time when offline. This accumulated time allows the stock market to run 50% faster when the game is opened again. This means that stock prices will update every ~4 seconds instead of 6.

1.2.13 Coding Contracts

Coding Contracts are a mechanic that lets players earn rewards in exchange for solving programming problems.

Coding Contracts are files with the “.cct” extensions. They can be accessed through the [Terminal](#) or through scripts using the [Coding Contract API](#)

Each contract has a limited number of attempts. If you provide the wrong answer too many times and exceed the number of attempts, the contract will self destruct (delete itself)

Currently, Coding Contracts are randomly generated and spawned over time. They can appear on any server (including your home computer), except for your purchased servers.

1.2.13.1 Running in Terminal

To run a Coding Contract in the Terminal, simply use the `run` command:

```
$ run some-contract.cct
```

Doing this will bring up a popup. The popup will display the contract’s problem, the number of attempts remaining, and an area to provide an answer.

1.2.13.2 Interacting through Scripts

See the [Coding Contract API](#).

1.2.13.3 Submitting Solutions

Different contract problem types will require different types of solutions. Some may be numbers, others may be strings or arrays. If a contract asks for a specific solution format, then use that. Otherwise, follow these rules when submitting solutions:

- String-type solutions should **not** have quotation marks surrounding the string (unless specifically asked for). Only quotation marks that are part of the actual string solution should be included.
- Array-type solutions should be submitted with each element in the array separated by commas. Brackets are optional. For example, both of the following are valid solution formats:

```
1, 2, 3  
[1, 2, 3]
```

However, if the solution is a multidimensional array, then all arrays that are not the outer-most array **DO** require the brackets. For example, an array of arrays can be submitted as one of the following:

```
[1, 2], [3, 4]  
[[1, 2], [3, 4]]
```

- Numeric solutions should be submitted normally, as expected

1.2.13.4 Rewards

There are currently four possible rewards for solving a Coding Contract:

- Faction Reputation for a specific Faction
- Faction Reputation for all Factions that you are a member of
- Company reputation for a specific Company
- Money

The ‘amount’ of reward varies based on the difficulty of the problem posed by the Coding Contract. There is no way to know what a Coding Contract’s exact reward will be until it is solved.

1.2.13.5 Notes

- The *scp* Terminal command does not work on Coding Contracts

1.2.13.6 List of all Problem Types

The following is a list of all of the problem types that a Coding Contract can contain. The list contains the name of (i.e. the value returned by `getContractType()`) and a brief summary of the problem it poses.

Name	Problem Summary
Find Largest Prime Factor	Given a number, find its largest prime factor. A prime factor is a factor that is a prime number.
Subarray with Maximum Sum	Given an array of integers, find the contiguous subarray (containing at least one number) which has the largest sum and return that sum.
Total Ways to Sum	Given a number, how many different distinct ways can that number be written as a sum of at least two positive integers?
Total Ways to Sum II	<p>You are given an array with two elements. The first element is an integer n.</p> <p>The second element is an array of numbers representing the set of available integers.</p> <p>How many different distinct ways can that number n be written as a sum of integers contained in the given set?</p> <p>You may use each integer in the set zero or more times.</p>
Spiralize Matrix	<p>Given an array of array of numbers representing a 2D matrix, return the elements of that matrix in clockwise spiral order.</p> <p>Example: The spiral order of</p> <pre> [1, 2, 3, 4] [5, 6, 7, 8] [9, 10, 11, 12] </pre> <p>is [1, 2, 3, 4, 8, 12, 11, 10, 9, 5, 6, 7]</p>
Array Jumping Game	<p>You are given an array of integers where each element represents the maximum possible jump distance from that position.</p> <p>For example, if you are at position i and your maximum jump length is n, then you can jump to any position from i to $i+n$.</p>
1.2. Basic Gameplay	<p>Assuming you are initially positioned at the start of the array, determine whether you are able to reach the last index of the array.</p>

1.3 Advanced Gameplay

This section documents Bitburner gameplay elements that are **not** immediately available and/or accessible to the player. These gameplay mechanics must be unlocked.

Warning: This page contains spoilers regarding the game's story/plot-line.

1.3.1 BitNodes

A BitNode is an important part of the game's storyline. In the game, you discover what BitNodes are by following the trail of clues left by the mysterious jump3r (essentially a minimal questline).

1.3.1.1 What is a BitNode

A BitNode is the complex simulated reality in which you reside. By following the messages from jump3r, you discover that humanity was enslaved by an advanced alien race, called the Enders, using virtual simulations that trapped the minds of humans.

However, the Enders didn't just create a single virtual reality to enslave humans, but many different simulations. In other words, there are many different BitNodes that exist. These BitNode are very different from each other.

jump3r tells you that the only hope for humanity is to destroy all of these BitNodes. Therefore, the end goal for the player is to enter and then destroy each BitNode at least once.

Destroying a BitNode resets most of the player's progress but grants the player a powerful second-tier persistent upgrade called a *Source-File*. Different BitNodes grant different Source-Files.

Each BitNode has unique characteristics that are related to varying backstories. For example, in one BitNode the world is in the middle of a financial catastrophe with a collapsing market. In this BitNode, most forms of income such as working at a company or Hacknet Nodes are significantly less profitable. Servers have less money on them and lowered growth rates, but it is easier to lower their security level using the `weaken()` Netscript function.

Furthermore, some BitNodes introduce new content and mechanics. For example there is one BitNode that grants access to the *Singularity API*. There is another BitNode in which you can manage a gang to earn money and reputation.

1.3.1.2 How to destroy a BitNode

Initially, the only way to destroy a BitNode is to join the Daedalus *Daedalus*. From Daedalus, the player can obtain an Augmentation called 'The Red Pill', which doesn't cost any money but does require a good amount of faction reputation.

After installing 'The Red Pill', the player must search for and then manually hack a server called 'w0rld_d43m0n'. This server requires a hacking level of 3000 in order to successfully hack it. This will destroy the player's current BitNode.

There is a second method of destroying a BitNode, but it must be unlocked by first destroying BitNode-6 or BitNode-7 (Bladeburners).

Todo: Link to Bladeburner documentation page here

When the player destroys a BitNode, most of his/her progress will be reset. This includes things such as Augmentations and RAM upgrades on the home computer. The only things that will persist through destroying BitNodes is:

- Source-Files
- Scripts on the home computer

Warning: This page contains spoilers for the game

1.3.2 Source-Files

Source-Files are a type of persistent upgrade that is more powerful than Augmentations. Source-Files are received by *destroying a BitNode*. There are many different BitNodes in the game and each BitNode will grant a different Source-File when it is destroyed.

A Source-File can be upgraded by destroying its corresponding BitNode a second or third time (AKA playing through that BitNode again). It can be upgraded to a maximum of level 3.

1.3.2.1 List of all Source-Files

BitNode-1: Source Genesis	<ul style="list-style-type: none"> * Let the player start with 32 GB of RAM on the home computer. * Increases all of the player's multipliers by 16%/24%/28%.
BitNode-2: Rise of the Underworld	<ul style="list-style-type: none"> * Let the player create Gangs in other BitNodes (although some BitNodes will disable this mechanic). * Increases the player's crime success rate, crime money, and charisma multipliers by 24%/36%/42%.
BitNode-3: Corporatocracy	<ul style="list-style-type: none"> * Let the player create Corporations in other BitNodes (although some BitNodes will disable this mechanic). * Increases the player's charisma and company salary multipliers by 8%/12%/14%.
BitNode-4: The Singularity	<ul style="list-style-type: none"> * Let the player access and use Netscript Singularity Functions in other BitNodes. * Each level of this Source-File reduces the RAM cost of singularity functions.
BitNode-5: Artificial Intelligence	<ul style="list-style-type: none"> * Unlocks <i>Intelligence</i>. * Unlocks <code>getBitNodeMultipliers()</code> and start with <code>Formulas.exe</code>. * Increases all of the player's hacking-related multipliers by 8%/12%/14%.
BitNode-6: Bladeburners	<ul style="list-style-type: none"> * Unlocks the Bladeburner feature in other BitNodes. * Increases all of the player's level and experience gain rate multipliers for combat stats by 8%/12%/14%.
BitNode-7: Bladeburners 2079	<ul style="list-style-type: none"> * Allows the player to access the <i>Bladeburner API</i> in other BitNodes. * Increases all of the player's Bladeburner multipliers by 8%/12%/14%.
1.3. Advanced Gameplay	
BitNode-8: Ghost of Wall Street	<ul style="list-style-type: none"> * Increases the player's hacking growth multiplier by 12%/18%/21%. * Level 1 grants permanent access to <i>WSE</i> and the <i>TIX API</i>.

1.3.3 Intelligence

Intelligence is a *stat* that is unlocked by having *Source-File 5* (i.e. Destroying BitNode-5).

Intelligence is unique because it is permanent and persistent. It never gets reset back to 1. However, gaining Intelligence experience is extremely slow. It is a stat that gradually builds up as you continue to play the game.

Intelligence will boost your production for many actions in the game, including:

- Hacking
- Infiltration
- Crime success rate
- Bladeburner
- Reputation gain for companies & factions
- Augmentation grafting speed

1.3.4 Sleeves

When VitaLife unveiled their Persona Core technology that allowed people to digitize and transfer their consciousness into other vessels, human bodies became nothing more than ‘sleeves’ for the human consciousness. This technology thus became known as “Sleeve technology”.

Sleeve technology unlocks two different gameplay features:

- Duplicate Sleeves
- Re-sleeving

Sleeve technology is unlocked in *BitNode-10*.

1.3.4.1 Duplicate Sleeves

Duplicate Sleeves are MK-V Synthoids (synthetic androids) into which your consciousness has been copied. In other words, these Synthoids contain a perfect duplicate of your mind.

Duplicate Sleeves are essentially clones which you can use to perform work-type actions, such as working for a company/faction or committing a crime. When sleeves perform these tasks, they will earn money, experience, and reputation.

Sleeves are their own individuals, which means they each have their own experience and stats.

When a sleeve earns experience, it earns experience for itself, the player’s original consciousness, as well as all of the player’s other sleeves.

Duplicate Sleeves are **not** reset when installing Augmentations, but they are reset when switching BitNodes.

1.3.4.1.1 Obtaining Duplicate Sleeves

There are two methods of obtaining Duplicate Sleeves:

1. Destroy BitNode-10. Each completion give you one additional Duplicate Sleeve
2. Purchase Duplicate Sleeves from *the faction The Covenant*. This is only available in BitNode-10. Sleeves purchased this way are **permanent** (they persist through BitNodes). You can purchase up to 5 Duplicate Sleeves from The Covenant.

1.3.4.1.2 Synchronization

Synchronization is a measure of how aligned your consciousness is with that of your Duplicate Sleeves. It is a numeral value between 1 and 100, and it affects how much experience is earned when the sleeve is performing a task.

Let N be the sleeve's synchronization. When the sleeve earns experience by performing a task, both the sleeve and the player's original host consciousness gain $N\%$ of the amount of experience normally earned by the task. All of the player's other sleeves earn $((N/100)^2 * 100)\%$ of the experience.

Synchronization can be increased by assigning sleeves to the 'Synchronize' task.

1.3.4.1.3 Sleeve Shock

Sleeve shock is a measure of how much trauma the sleeve has due to being placed in a new body. It is a numeral value between 0 and 99, where 99 indicates full shock and 0 indicates no shock. Shock affects the amount of experience earned by the sleeve.

Sleeve shock slowly decreases over time. You can further increase the rate at which it decreases by assigning sleeves to the 'Shock Recovery' task.

1.3.4.1.4 Augmentations

You can purchase *Augmentations* for your Duplicate Sleeves. In order to do this, the Sleeve's Shock must be at 0. Any Augmentation that is currently available to you through a faction is also available for your Duplicate Sleeves. There are a few Augmentations, such as NeuroFlux Governor and Bladeburner-specific ones, that cannot be purchased for a Duplicate Sleeve.

When you purchase an Augmentation for a Duplicate Sleeve, it is instantly installed. When this happens, the Sleeve's stats are instantly reset back to 0, similar to when you normally install Augmentations.

The cost of purchasing an Augmentation for a Duplicate Sleeve is **not** affected by how many Augmentations you have purchased for yourself, and vice versa.

1.3.4.1.5 Memory

Sleeve memory dictates what a sleeve's synchronization will be when its reset by switching BitNodes. For example, if a sleeve has a memory of 10, then when you switch BitNodes its synchronization will initially be set to 10, rather than 1.

Memory can only be increased by purchasing upgrades from The Covenant. Just like the ability to purchase additional sleeves, this is only available in BitNode-10.

Memory is a persistent stat, meaning it never gets reset back to 1. The maximum possible value for a sleeve's memory is 100.

1.3.5 Grafting

Grafting is an experimental process through which you can obtain the benefits of Augmentations, without needing to reboot your body.

Grafting can be done at VitaLife in New Tokyo, where you'll find a shady researcher with questionable connections. From there, you can spend a sum of money to begin grafting Augmentations. This will take some time. When done, the Augmentation will be applied to your character without needing to install.

Be warned, some who have tested grafting have reported an unidentified malware. Dubbed “Entropy”, this virus seems to grow in potency as more Augmentations are grafted, causing unpredictable affects to the victim.

Note that when crafting an Augmentation, cancelling will **not** save your progress, and the money spent will **not** be returned.

1.3.6 Hacking algorithms

There are three primary families of hacking algorithms. This guide will go over each of them and advise on how they can be implemented.

1.3.6.1 Self-contained algorithms

Difficulty: Easy

Pros:

- Easy to implement
- Does not require other scripts to work
- Works at any stage of the game

Cons:

- Limits income generation
- Extremely RAM inefficient
- Utilizes script online time poorly
- Risk of over hacking

Self-contained algorithms are the simplest family of hacking algorithms to implement. Each script is tasked with choosing which function to execute based on the status of the target server. Because of this, they guarantee a consistent, but relatively small, flow of money.

The general logic goes like this:

```
loop forever {
  if security is not minimum {
    weaken(target)
  } else if money is not maximum {
    grow(target)
  } else {
    hack(target)
  }
}
```

This algorithm is perfectly capable of paving the way through the majority of the game, but it has a few significant issues.

- It tends to make all your scripts on every server do the same thing. (e.g. If the target is 0.01 security above the minimum, all scripts will decide to weaken, when only a handful of threads should be devoted to the task)
- At higher thread counts, these scripts have the potential to hack the server to \$0, or maximum security, requiring a long setup time while the scripts return the server to the best stats.
- Requires function calls such as *getServerSecurityLevel()* and *getServerMoneyAvailable()*, as well as calling all three hacking functions, increasing RAM cost which is multiplied by the number of allocated threads

1.3.6.2 Loop algorithms

Difficulty: Easy to Medium

Pros:

- Simple to understand
- Works at any stage of the game
- Maximize RAM usage

Cons:

- Requires a script that handles deployment

By splitting our hack, weaken, and grow functions into three separate scripts, we can both remove our reliance on functions such as `getServerSecurityLevel()` as well as removing functions that cannot work concurrently, reducing RAM requirements, and thus increasing our thread limits. Loop scripts are formatted like this:

```
loop forever {
    hack(target) // or grow, or weaken
}
```

Now we can take the total amount of threads available and split it and allocate, for example:

- 1 part to the hack scripts
- 10 parts to the grow scripts
- 2 parts to the weaken scripts

Meaning if we have space for 100 threads across the entire network 7 threads will go to the hack scripts, 76 threads will go to the grow scripts and 15 threads will go to the weaken scripts. The ratios described here are arbitrary and can be greatly improved through the use of the analyze functions, and later, through the use of Formulas.exe.

When utilizing this strategy, monitor the amount of money and security on the target server, if the money is not hovering around maximum and the security around the minimum, the ratios should be tweaked until that is the case.

Utilizing `sleep()` or `asleep()` to ensure that your scripts do not all start at the same time can decrease the chance of issues associated with overhacking occurring. Both functions have a ram cost of zero.

1.3.6.3 Batch algorithms (HGW, HWGW, or Cycles)

Difficulty: Hard

Pros:

- Maximum potential income

Cons:

- Very difficult to implement without prior programming knowledge
- Very difficult to make work on servers with less than 1TB of RAM

Batch algorithms utilize a master script that uses `exec()` many scripts which utilize a relevant hacking function in batches.

The scripts used to execute the hacking functions are even simpler than the previous algorithms but a complex controller is required to calculate the effect, time taken, and the necessary delay.

```
sleep(a bit)
hack(target) // or grow, or weaken
```

A few things need to be known before this algorithm can be implemented:

- The effects of hack and grow depend on the server security level, a higher security level results in a reduced effect. You only want these effects to occur when the security level is minimized.
- The time taken to execute hack, grow, or weaken is determined when the function is called and is based on the security level of the target server and your hacking level. You only want these effects to start when the security level is minimized.
- The effects of hack, grow, and weaken, are determined when the time is completed, rather than at the beginning. Hack should finish when security is minimum and money is maximum. Grow should finish when security is minimum, shortly after a hack occurred. Weaken should occur when security is not at a minimum due to a hack or grow increasing it.

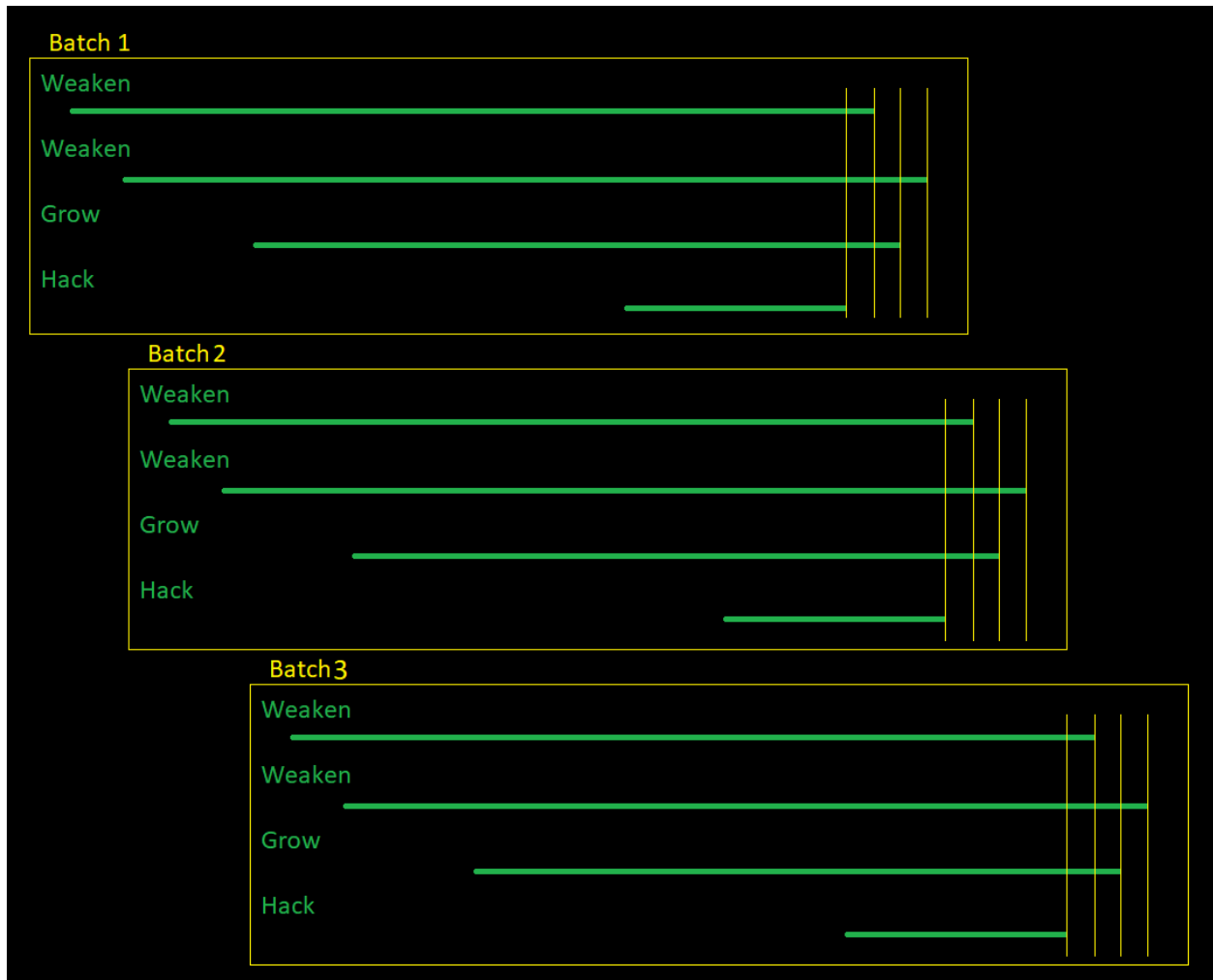
A single batch consists of four actions:

1. A hack script removes a predefined, precalculated amount of money from the target server.
2. A weaken script counters the security increase of the hack script.
3. A grow script counters the money decrease caused by the hack script.
4. A weaken script counters the security increase caused by the grow script.

It is also important that these 4 scripts finish in the order specified above, and all of their effects be precalculated to optimize the ratios between them. This is the reason for the delay in the scripts.

It is possible to create batches with 3 scripts (HGW) but the efficiency of grow will be harmed by the security increase caused by the hack scripts.

The following is an image demonstrating batches in action:



Batches only function predictably when the target server is at minimum security and maximum money, so your script must also handle preparing a server for your batches. You can utilize batches to prepare a server by using no hack threads during preparation.

Depending on your computer's performance as well as a few other factors, the necessary delay between script execution times may range between 20ms and 200ms, you want to fine-tune this value to be as low as possible while also avoiding your scripts finishing out of order. Anything lower than 20ms will not work due to javascript limitations.

1.4 Keyboard Shortcuts

This page documents the various keyboard shortcuts that can be used in the game.

1.4.1 Game Navigation

These are used to switch between the different menus/tabs in the game. These shortcuts are almost always available. Exceptions include:

- Working at a company or for a faction
- Creating a program

- Taking a university class
- Training at a gym
- Active Mission (aka Hacking Mission)

Shortcut	Action
Alt + t	Switch to <i>Terminal</i>
Alt + c	Switch to 'Stats' page
Alt + e	Switch to Script Editor. Will open up the last-edited file or a new file
Alt + s	Switch to 'Active Scripts' page
Alt + h	Switch to 'Hacknet Nodes' page
Alt + w	Switch to 'City' page
Alt + j	Go to the company where you are employed ('Job' page on navigation menu)
Alt + r	Go to Travel Agency in current City ('Travel' page on navigation menu)
Alt + p	Switch to 'Create Program' page
Alt + f	Switch to 'Factions' page
Alt + a	Switch to 'Augmentations' page
Alt + u	Switch to 'Tutorial' page
Alt + o	Switch to 'Options' page
Alt + g	Switch to 'Gang' page
Alt + b	Switch to 'Bladeburner' page

1.4.2 Terminal Shortcuts

These shortcuts are available only in the *Terminal*

Shortcut	Action
Up/Down arrow	Cycle through previous commands
Ctrl + c	Cancel a hack/analyze action
Ctrl + l	Clear screen
Tab	Autocomplete command

1.4.3 Terminal Bash Shortcuts

These shortcuts were implemented to better emulate a bash shell. They must be enabled in your *Terminal*'s *.fconf* file. This can be done by entering the Terminal command:

```
nano .fconf
```

and then setting the *ENABLE_BASH_HOTKEYS* option to 1.

Note that these Bash shortcuts override any other shortcuts defined in the game (unless otherwise noted), as well as your browser's shortcuts

Also note that more Bash-like shortcuts will be implemented in the future

Shortcut	Action
Ctrl + c	Clears current Terminal input (does NOT override default Ctrl + c command)
Ctrl + p	Same as Up Arrow
Ctrl + m	Same as Down Arrow
Ctrl + a	Move cursor to beginning of line (same as 'Home' key)
Ctrl + e	Move cursor to end of line (same as 'End' key)
Ctrl + b	Move cursor to previous character
Alt + b	Move cursor to previous word
Ctrl + f	Move cursor to next character
Alt + f	Move cursor to next word
Ctrl + h/d	Delete previous character ('Backspace')

1.4.4 Popup/Dialog Box Shortcuts

The following shortcuts work if there are any popup or dialog boxes on the screen.

Shortcut	Action
Esc	Close the current popup, cancelling any prompts on a dialog box
Enter	Clicks the “Yes/Confirm” option for every dialog box

1.5 Game Frozen or Stuck?

1.5.1 Infinite Loop in NetscriptJS

If your game is frozen or stuck in any way, then the most likely culprit is an infinitely running loop in [NS2](#). To get past the freezing, run the game with *?noScripts* in the URL:

<https://danielyxie.github.io/bitburner/?noScripts>

Then, to fix your script, make sure you have a sleep or any other timed function like *hack()* or *grow()* in any infinite loops:

```
while(true) {
  // This is an infinite loop that does something
  ...
  await ns.sleep(1000); // Add a 1s sleep to prevent freezing
}
```

1.5.2 Bug

Otherwise, the game is probably frozen/stuck due to a bug. To report a bug, follow the guidelines [here](#).

1.6 Guides & Tips

Getting Started Guide for Intermediate Programmers

Beginners FAQ

1.6.1 Getting Started Guide for Beginner Programmers

Note: Note that the scripts and strategies given in this guide aren't necessarily optimal. They're just meant to introduce you to the game and help you get started.

This is an introductory guide to getting started with Bitburner. It is not meant to be a comprehensive guide for the entire game, only the early stages. If you are confused or overwhelmed by the game, especially the programming and scripting aspects, this guide is perfect for you!

Note that this guide is tailored towards those with minimal programming experience.

1.6.1.1 Introduction

Bitburner is a cyberpunk-themed incremental RPG. The player progresses by raising their *Stats*, earning money, and *climbing the corporate ladder*. Eventually, after reaching certain criteria, the player will begin receiving invitations from *Factions*. Joining these factions and working for them will unlock *Augmentations*. Purchasing and installing Augmentations provide persistent upgrades and are necessary for progressing in the game.

The game has a minimal story/quest-line that can be followed to reach the end of the game. Since this guide is only about getting started with Bitburner, it will not cover the entire “quest-line”.

1.6.1.2 First Steps

I'm going to assume you followed the introductory tutorial when you first began the game. In this introductory tutorial you created a script called `n00dles.script` and ran it on the `n00dles` server. Right now, we'll kill this script. There are two ways to do this:

1. You can go to the Terminal and enter:

```
$ kill n00dles.script
```

2. You can go to the Active Scripts page (*Keyboard shortcut* Alt + s) and press the “Kill Script” button for `n00dles.script`.

If you skipped the introductory tutorial, then ignore the part above. Instead, go to the Hacknet Nodes page (*Keyboard shortcut* Alt + h) and purchase a Hacknet Node to start generating some passive income.

1.6.1.3 Creating our First Script

Now, we'll create a generic hacking script that can be used early on in the game (or throughout the entire game, if you want).

Before we write the script, here are some things you'll want to familiarize yourself with:

- *General Hacking Mechanics*
- *Server Security*
- `hack()`
- `grow()`
- `weaken()`
- `brutesh()`
- `nuke()`

To briefly summarize the information from the links above: Each server has a security level that affects how difficult it is to hack. Each server also has a certain amount of money, as well as a maximum amount of money it can hold. Hacking a server steals a percentage of that server's money. The `hack()` Netscript function is used to hack a server. The `grow()` Netscript function is used to increase the amount of money available on a server. The `weaken()` Netscript function is used to decrease a server's security level.

Now let's move on to actually creating the script. Go to your home computer and then create a script called `early-hack-template.script` by going to Terminal and entering the following two commands:

```
$ home
$ nano early-hack-template.script
```

This will take you to the script editor, which you can use to code and create *Scripts*. It will be helpful to consult the *Netscript* documentation. Specifically, you'll want to take a look at *Netscript Basic Functions*.

Enter the following code in the script editor:

```
// Defines the "target server", which is the server
// that we're going to hack. In this case, it's "n00dles"
var target = "n00dles";

// Defines how much money a server should have before we hack it
// In this case, it is set to 75% of the server's max money
var moneyThresh = getServerMaxMoney(target) * 0.75;

// Defines the maximum security level the target server can
// have. If the target's security level is higher than this,
// we'll weaken it before doing anything else
var securityThresh = getServerMinSecurityLevel(target) + 5;

// If we have the BruteSSH.exe program, use it to open the SSH Port
// on the target server
if (fileExists("BruteSSH.exe", "home")) {
    brutessh(target);
}

// Get root access to target server
nuke(target);

// Infinite loop that continuously hacks/grows/weakens the target server
while(true) {
    if (getServerSecurityLevel(target) > securityThresh) {
        // If the server's security level is above our threshold, weaken it
        weaken(target);
    } else if (getServerMoneyAvailable(target) < moneyThresh) {
        // If the server's money is less than our threshold, grow it
        grow(target);
    } else {
        // Otherwise, hack it
        hack(target);
    }
}
```

The script above contains comments that document what it does, but let's go through it step-by-step anyways.

```
var target = "n00dles";
```

This first command defines a string which contains our target server. That's the server that we're going to hack. For now, it's set to `n00dles` because that's the only server with a required hacking level of 1. If you want to hack a different

server, simply change this variable to be the hostname of another server.

```
var moneyThresh = getServerMaxMoney(target) * 0.75;
```

This second command defines a numerical value representing the minimum amount of money that must be available on the target server in order for our script to hack it. If the money available on the target server is less than this value, then our script will *grow()* the server rather than hacking it. It is set to 75% of the maximum amount of money that can be available on the server. The *getServerMaxMoney()* Netscript function is used to find this value

```
var securityThresh = getServerMinSecurityLevel(target) + 5;
```

This third command defines a numerical value representing the maximum security level the target server can have. If the target server's security level is higher than this value, then our script will *weaken()* the script before doing anything else.

```
if (fileExists("BruteSSH.exe", "home")) {  
    brutessh(target);  
}  
  
nuke(target);
```

This section of code is used to gain root access on the target server. This is necessary for hacking. See [here for more details](#).

```
while (true) {  
    if (getServerSecurityLevel(target) > securityThresh) {  
        // If the server's security level is above our threshold, weaken it  
        weaken(target);  
    } else if (getServerMoneyAvailable(target) < moneyThresh) {  
        // Otherwise, if the server's money is less than our threshold, grow it  
        grow(target);  
    } else {  
        // Otherwise, hack it  
        hack(target);  
    }  
}
```

This is the main section that drives our script. It dictates the script's logic and carries out the hacking operations. The *while (true)* creates an infinite loop that will continuously run the hacking logic until the the script is killed.

1.6.1.4 Running our Scripts

Now we want to start running our hacking script so that it can start earning us money and experience. Our home computer only has 8GB of RAM and we'll be using it for something else later. So instead, we'll take advantage of the RAM on other machines.

Go to Terminal and enter the following command:

```
$ scan-analyze 2
```

This will show detailed information about some servers on the network. The **network is randomized so it will be different for every person**. Here's what mine showed at the time I made this:

```
[home ~]> scan-analyze 2  
~~~~~ Beginning scan-analyze ~~~~~
```

(continues on next page)

(continued from previous page)

```
n00dles
--Root Access: YES, Required hacking skill: 1
--Number of open ports required to NUKE: 0
--RAM: 4.00GB

----zer0
-----Root Access: NO, Required hacking skill: 75
-----Number of open ports required to NUKE: 1
-----RAM: 32.00GB

foodnstuff
--Root Access: NO, Required hacking skill: 1
--Number of open ports required to NUKE: 0
--RAM: 16.00GB

sigma-cosmetics
--Root Access: NO, Required hacking skill: 5
--Number of open ports required to NUKE: 0
--RAM: 16.00GB

joesguns
--Root Access: NO, Required hacking skill: 10
--Number of open ports required to NUKE: 0
--RAM: 16.00GB

----max-hardware
-----Root Access: NO, Required hacking skill: 80
-----Number of open ports required to NUKE: 1
-----RAM: 32.00GB

----CSEC
-----Root Access: NO, Required hacking skill: 54
-----Number of open ports required to NUKE: 1
-----RAM: 8.00GB

hong-fang-tea
--Root Access: NO, Required hacking skill: 30
--Number of open ports required to NUKE: 0
--RAM: 16.00GB

----nectar-net
-----Root Access: NO, Required hacking skill: 20
-----Number of open ports required to NUKE: 0
-----RAM: 16.00GB

harakiri-sushi
--Root Access: NO, Required hacking skill: 40
--Number of open ports required to NUKE: 0
--RAM: 16.00GB

iron-gym
--Root Access: NO, Required hacking skill: 100
--Number of open ports required to NUKE: 1
--RAM: 32.00GB
```

Take note of the following servers:

- sigma-cosmetics

- joesguns
- nectar-net
- hong-fang-tea
- harakiri-sushi

All of these servers have 16GB of RAM. Furthermore, all of these servers do not require any open ports in order to NUKÉ. In other words, we can gain root access to all of these servers and then run scripts on them.

First, let's determine how many threads of our hacking script we can run. [Read more about multithreading scripts here](#) The script we wrote uses 2.6GB of RAM. You can check this using the following Terminal command:

```
$ mem early-hack-template.script
```

This means we can run 6 threads on a 16GB server. Now, to run our scripts on all of these servers, we have to do the following:

1. Use the `scp` Terminal command to copy our script to each server.
2. Use the `connect` Terminal command to connect to a server.
3. Use the `run` Terminal command to run the `NUKE.exe` program and gain root access.
4. Use the `run` Terminal command again to run our script.
5. Repeat steps 2-4 for each server.

Here's the sequence of Terminal commands I used in order to achieve this:

```
$ home
$ scp early-hack-template.script n00dles
$ scp early-hack-template.script sigma-cosmetics
$ scp early-hack-template.script joesguns
$ scp early-hack-template.script nectar-net
$ scp early-hack-template.script hong-fang-tea
$ scp early-hack-template.script harakiri-sushi
$ connect n00dles
$ run NUKE.exe
$ run early-hack-template.script -t 1
$ home
$ connect sigma-cosmetics
$ run NUKE.exe
$ run early-hack-template.script -t 6
$ home
$ connect joesguns
$ run NUKE.exe
$ run early-hack-template.script -t 6
$ home
$ connect hong-fang-tea
$ run NUKE.exe
$ run early-hack-template.script -t 6
$ home
$ connect harakiri-sushi
$ run NUKE.exe
$ run early-hack-template.script -t 6
$ home
$ connect hong-fang-tea
$ connect nectar-net
$ run NUKE.exe
$ run early-hack-template.script -t 6
```

Note: Pressing the `Tab` key in the middle of a Terminal command will attempt to auto-complete the command. For example, if you type in `scp ea` and then hit `Tab`, the rest of the script's name should automatically be filled in. This works for most commands in the game!

The `home` Terminal command is used to connect to the home computer. When running our scripts with the `run early-hack-template.script -t 6` command, the `-t 6` specifies that the script should be run with 6 threads.

Note that the `nectar-net` server isn't in the home computer's immediate network. This means you can't directly connect to it from home. You will have to search for it inside the network. The results of the `scan-analyze 2` command we ran before will show where it is. In my case, I could connect to it by going from `hong-fang-tea -> nectar-net`. However, this will probably be different for you.

After running all of these Terminal commands, our scripts are now up and running. These will earn money and hacking experience over time. These gains will be really slow right now, but they will increase once our hacking skill rises and we start running more scripts.

1.6.1.5 Increasing Hacking Level

There are many servers besides `n00dles` that can be hacked, but they have higher required hacking levels. Therefore, we should raise our hacking level. Not only will this let us hack more servers, but it will also increase the effectiveness of our hacking against `n00dles`.

The easiest way to train your hacking level is to visit Rothman University. You can do this by clicking the `City` tab on the left-hand navigation menu, or you can use the *keyboard shortcut* `Alt + w`. Rothman University should be one of the buttons near the top. Click the button to go to the location.

Once you go to Rothman University, you should see a screen with several options. These options describe different courses you can take. You should click the first button, which says: "Study Computer Science (free)".

After you click the button, you will start studying and earning hacking experience. While you are doing this, you cannot interact with any other part of the game until you click the button that says "Stop taking course".

Right now, we want a hacking level of 10. You need approximately 174 hacking experience to reach level 10. You can check how much hacking experience you have by clicking the `Stats` tab on the left-hand navigation menu, or by using *Keyboard shortcut* `Alt + c`. Since studying at Rothman University earns you 1 experience per second, this will take 174 seconds, or approximately 3 minutes. Feel free to do something in the meantime!

1.6.1.6 Editing our Hacking Script

Now that we have a hacking level of 10, we can hack the `joesguns` server. This server will be slightly more profitable than `n00dles`. Therefore, we want to change our hacking script to target `joesguns` instead of `n00dles`.

Go to Terminal and edit the hacking script by entering:

```
$ home
$ nano early-hack-template.script
```

At the top of the script, change the `target` variable to be `joesguns`:

```
var target = "joesguns";
```

Note that this will **NOT** affect any instances of the script that are already running. This will only affect instances of the script that are ran from this point forward.

1.6.1.7 Creating a New Script to Purchase New Servers

Next, we're going to create a script that automatically purchases additional servers. These servers will be used to run many scripts. Running this script will initially be very expensive since purchasing a server costs money, but it will pay off in the long run.

In order to create this script, you should familiarize yourself with the following Netscript functions:

- `purchaseServer()`
- `getPurchasedServerCost()`
- `getPurchasedServerLimit()`
- `getServerMoneyAvailable()`
- `scp()`
- `exec()`

Create the script by going to Terminal and typing:

```
$ home
$ nano purchase-server-8gb.script
```

Paste the following code into the script editor:

```
// How much RAM each purchased server will have. In this case, it'll
// be 8GB.
var ram = 8;

// Iterator we'll use for our loop
var i = 0;

// Continuously try to purchase servers until we've reached the maximum
// amount of servers
while (i < getPurchasedServerLimit()) {
    // Check if we have enough money to purchase a server
    if (getServerMoneyAvailable("home") > getPurchasedServerCost(ram)) {
        // If we have enough money, then:
        // 1. Purchase the server
        // 2. Copy our hacking script onto the newly-purchased server
        // 3. Run our hacking script on the newly-purchased server with 3 threads
        // 4. Increment our iterator to indicate that we've bought a new server
        var hostname = purchaseServer("pserv-" + i, ram);
        scp("early-hack-template.script", hostname);
        exec("early-hack-template.script", hostname, 3);
        ++i;
    }
}
```

This code uses a while loop to purchase the maximum amount of servers using the `purchaseServer()` Netscript function. Each of these servers will have 8GB of RAM, as defined in the `ram` variable. Note that the script uses the command `getServerMoneyAvailable("home")` to get the amount of money you currently have. This is then used to check if you can afford to purchase a server.

Whenever the script purchases a new server, it uses the `scp()` function to copy our script onto that new server, and then it uses the `exec()` function to execute it on that server.

To run this script, go to Terminal and type:


```
$ run purchase-server-8gb.script
```

This purchase will continuously run until it has purchased the maximum number of servers. When this happens, it'll mean that you have a bunch of new servers that are all running hacking scripts against the `joesguns` server!

Note: The reason we're using so many scripts to hack `joesguns` instead of targeting other servers is because it's more effective. This early in the game, we don't have enough RAM to efficiently hack multiple targets, and trying to do so would be slow as we'd be spread too thin. You should definitely do this later on, though!

Note that purchasing a server is fairly expensive, and purchasing the maximum amount of servers even more so. At the time of writing this guide, the script above requires \$11 million in order to finish purchasing all of the 8GB servers. Therefore, we need to find additional ways to make money to speed up the process! These are covered in the next section.

1.6.1.8 Additional Sources of Income

There are other ways to gain money in this game besides scripts & hacking.

1.6.1.8.1 Hacknet Nodes

If you completed the introductory tutorial, you were already introduced to this method: Hacknet Nodes. Once you have enough money, you can start upgrading your Hacknet Nodes in order to increase your passive income stream. This is completely optional. Since each Hacknet Node upgrade takes a certain amount of time to "pay itself off", it may not necessarily be in your best interest to use these.

Nonetheless, Hacknet Nodes are a good source of income early in the game, although their effectiveness tapers off later on. If you do wind up purchasing and upgrading Hacknet Nodes, I would suggest only upgrading their levels for now. I wouldn't bother with RAM and Core upgrades until later on.

1.6.1.8.2 Crime

The best source of income right now is from *committing crimes*. This is because it not only gives you a large amount of money, but it also raises your hacking level. To commit crimes, click on the `City` tab on the left-hand navigation menu or use the *Keyboard shortcut* `Alt + w`. Then, click on the link that says `The Slums`.

In the Slums, you can attempt to commit a variety of crimes, each of which gives certain types of experience and money if successful. See *Crimes* for more details.

Note: You are not always successful when you attempt to commit a crime. Nothing bad happens if you fail a crime, but you won't earn any money and the experience gained will be reduced. Raising your stats improves your chance of successfully committing a crime.

Right now, the best option is the `Rob Store` crime. This takes 60 seconds to attempt and gives \$400k if successful. I suggest this crime because you don't have to click or check in too often since it takes a whole minute to attempt. Furthermore, it gives hacking experience, which is very important right now.

Alternatively, you can also use the `Shoplift` crime. This takes 2 seconds to attempt and gives \$15k if successful. This crime is slightly easier and is more profitable than `Rob Store`, but it requires constant clicking and it doesn't give hacking experience.

1.6.1.8.3 Work for a Company

If you don't want to constantly check in on the game to commit crimes, there's another option that's much more passive: working for a *company*. This will not be nearly as profitable as crimes, but it's completely passive.

Go to the `City` tab on the left-hand navigation menu and then go to `Joe's Guns`. At `Joe's Guns`, there will be an option that says `Apply to be an Employee`. Click this to get the job. Then, a new option will appear that simply says `Work`. Click this to start working. Working at `Joe's Guns` earns \$110 per second and also grants some experience for every stat except hacking.

Working for a company is completely passive. You can choose to focus on your work, do something else simultaneously, or switch between those two. While you focus on work, you will not be able to do anything else in the game. If you do something else meanwhile, you will not gain reputation at the same speed. You can cancel working at any time. You'll notice that cancelling your work early causes you to lose out on some reputation gains, but you shouldn't worry about this. Company reputation isn't important right now.

Once your hacking hits level 75, you can visit `Carmichael Security` in the city and get a software job there. This job offers higher pay and also earns you hacking experience.

There are many more companies in the `City` tab that offer more pay and also more gameplay features. Feel free to explore!

1.6.1.9 After you Purchase your New Servers

After you've made a total of \$11 million, your automatic server-purchasing script should finish running. This will free up some RAM on your home computer. We don't want this RAM to go to waste, so we'll make use of it. Go to `Terminal` and enter the following commands:

```
$ home
$ run early-hack-template.script -t 3
```

1.6.1.10 Reaching a Hacking Level of 50

Once you reach a hacking level of 50, two new important parts of the game open up.

1.6.1.10.1 Creating your first program: `BruteSSH.exe`

On the left-hand navigation menu you will notice a `Create Programs` tab with a red notification icon. This indicates that there are programs available to be created. Click on that tab (or use *Keyboard shortcut* `Alt + p`) and you'll see a list of all the programs you can currently create. Hovering over a program will give a brief description of its function. Simply click on a program to start creating it.

Right now, the program we want to create is `BruteSSH.exe`. This program is used to open up SSH ports on servers. This will allow you to hack more servers, as many servers in the game require a certain number of opened ports in order for `NUKE.exe` to gain root access.

When you are creating a program, you cannot interact with any other part of the game. Feel free to cancel your work on creating a program at any time, as your progress will be saved and can be picked back up later. `BruteSSH.exe` takes about 10 minutes to complete.

1.6.1.10.2 Optional: Create AutoLink.exe

On the `Create Programs` page, you will notice another program you can create called `AutoLink.exe`. If you don't mind waiting another 10-15 minutes, you should go ahead and create this program. It makes it much less tedious to connect to other servers, but it's not necessary for progressing.

1.6.1.10.3 Joining your first faction: CyberSec

Shortly after you reached level 50 hacking, you should have received a message that said this:

```
Message received from unknown sender:

We've been watching you. Your skills are very impressive. But you're wasting
your talents. If you join us, you can put your skills to good use and change
the world for the better. If you join us, we can unlock your full potential.
But first, you must pass our test. Find and hack our server using the Terminal.

-CyberSec

This message was saved as csec-test.msg onto your home computer.
```

If you didn't, or if you accidentally closed it, that's okay! Messages get saved onto your home computer. Enter the following Terminal commands to view the message:

```
$ home
$ cat csec-test.msg
```

This message is part of the game's main "quest-line". It is a message from the `CyberSec` faction that is asking you to pass their test. Passing their test is simple, you just have to find their server and hack it through the Terminal. Their server is called `CSEC`. To do this, we'll use the `scan-analyze` Terminal command, just like we did before:

```
$ home
$ scan-analyze 2
```

This will show you the network for all servers that are up to 2 "nodes" away from your home computer. Remember that the network is randomly generated so it'll look different for everyone. Here's the relevant part of my `scan-analyze` results:

```
>iron-gym
--Root Access: NO, Required hacking skill: 100
--Number of open ports required to NUKE: 1
--RAM: 32

---->zer0
-----Root Access: NO, Required hacking skill: 75
-----Number of open ports required to NUKE: 1
-----RAM: 32

---->CSEC
-----Root Access: NO, Required hacking skill: 54
-----Number of open ports required to NUKE: 1
-----RAM: 8
```

This tells me that I can reach `CSEC` by going through `iron-gym`:

```
$ connect iron-gym
$ connect CSEC
```

Note: If you created the `AutoLink.exe` program earlier, then there is an easier method of connecting to CSEC. You'll notice that in the `scan-analyze` results, all of the server hostnames are white and underlined. You can simply click one of the server hostnames in order to connect to it. So, simply click CSEC!

Note: Make sure you notice the required hacking skill for the CSEC server. This is a random value between 51 and 60. Although you receive the message from CSEC once you hit 50 hacking, you cannot actually pass their test until your hacking is high enough to install a backdoor on their server.

After you are connected to the CSEC server, you can backdoor it. Note that this server requires one open port in order to gain root access. We can open the SSH port using the `BruteSSH.exe` program we created earlier. In Terminal:

```
$ run BruteSSH.exe
$ run NUKE.exe
$ backdoor
```

After you successfully install the backdoor, you should receive a faction invitation from CyberSec shortly afterwards. Accept it. If you accidentally reject the invitation, that's okay. Just go to the `Factions` tab (*Keyboard shortcut* Alt + f) and you should see an option that lets you accept the invitation.

Congrats! You just joined your first faction. Don't worry about doing anything with this faction yet, we can come back to it later.

1.6.1.11 Using Additional Servers to Hack Joesguns

Once you have the `BruteSSH.exe` program, you will be able to gain root access to several additional servers. These servers have more RAM that you can use to run scripts. We'll use the RAM on these servers to run more scripts that target joesguns.

1.6.1.11.1 Copying our Scripts

The server's we'll be using to run our scripts are:

- neo-net
- zer0
- max-hardware
- iron-gym

All of these servers have 32GB of RAM. You can use the Terminal command `scan-analyze 3` to see for yourself. To copy our hacking scripts onto these servers, go to Terminal and run:

```
$ home
$ scp early-hack-template.script neo-net
$ scp early-hack-template.script zer0
$ scp early-hack-template.script max-hardware
$ scp early-hack-template.script iron-gym
```

Since each of these servers has 32GB of RAM, we can run our hacking script with 12 threads on each server. By now, you should know how to connect to servers. So find and connect to each of the servers above using the `scan-analyze 3` Terminal command. Then, use following Terminal command to run our hacking script with 12 threads:

```
$ run early-hack-template.script -t 12
```

Remember that if you have the `AutoLink.exe` program, you can simply click on the hostname of a server after running `scan-analyze` to connect to it.

1.6.1.12 Profiting from Scripts & Gaining Reputation with CyberSec

Now it's time to play the waiting game. It will take some time for your scripts to start earning money. Remember that most of your scripts are targeting `joesguns`. It will take a bit for them to `grow()` and `weaken()` the server to the appropriate values before they start hacking it. Once they do, however, the scripts will be very profitable.

Note: For reference, in about two hours after starting my first script, my scripts had a production rate of \$20k per second and had earned a total of \$70 million. (You can see these stats on the `Active Scripts` tab).

After another 15 minutes, the production rate had increased to \$25k per second and the scripts had made an additional \$55 million.

Your results will vary based on how fast you earned money from crime/working/hacknet nodes, but this will hopefully give you a good indication of how much the scripts can earn.

In the meantime, we are going to be gaining reputation with the `CyberSec faction`. Go to the `Factions` tab on the left-hand navigation menu, and from there select `CyberSec`. In the middle of the page there should be a button for `Hacking Contracts`. Click it to start earning reputation for the `CyberSec` faction (as well as some hacking experience). The higher your hacking level, the more reputation you will gain. Note that while you are working for a faction, you can choose to not interact with the rest of the game in any way to gain reputation at full speed. You can also select to do something else simultaneously, gaining reputation a bit more slowly, until you focus again. You can cancel your faction work at any time with no penalty to your reputation gained so far.

1.6.1.13 Purchasing Upgrades and Augmentations

As I mentioned before, within 1-2 hours I had earned over \$200 million. Now, it's time to spend all of this money on some persistent upgrades to help progress!

1.6.1.13.1 Upgrading RAM on Home computer

The most important thing to upgrade right now is the RAM on your home computer. This will allow you to run more scripts.

To upgrade your RAM, go to the `City` tab and visit the company `Alpha Enterprises`. There will be an option that says `Purchase additional RAM for Home Computer`. Click it and follow the dialog box to upgrade your RAM.

I recommend getting your home computer's RAM to *at least* 128GB. Getting it even higher would be better.

1.6.1.13.2 Purchasing your First Augmentations

Once you get ~1000 reputation with the `CyberSec faction`, you can purchase your first `Augmentation` from them.

To do this, go to the `Factions` tab on the left-hand navigation menu (*Keyboard shortcut* `Alt + f`) and select `CyberSec`. There is an button near the bottom that says `Purchase Augmentations`. This will bring up a page that displays all of the `Augmentations` available from `CyberSec`. Some of them may be locked right now. To unlock these, you will need to earn more reputation with `CyberSec`.

`Augmentations` give persistent upgrades in the form of multipliers. These aren't very powerful early in the game because the multipliers are small. However, the effects of `Augmentations` stack multiplicatively **with each other**, so as you continue to install many `Augmentations` their effects will increase significantly.

Because of this, I would recommend investing more in `RAM` upgrades for your home computer rather than `Augmentations` early on. Having enough `RAM` to run many scripts will allow you to make much more money, and then you can come back later on and get all these `Augmentations`.

Right now, I suggest purchasing at the very least the `Neurotrainer I` `Augmentation` from `CyberSec`. If you have the money to spare, I would also suggest getting `BitWire` and several levels of the `NeuroFlux Governor` `Augmentations`. Note that each time you purchase an `Augmentation`, *the price of purchasing another increases by 90%*, so make sure you buy the most expensive `Augmentation` first. Don't worry, once you choose to install `Augmentations`, their prices will reset back to their original values.

1.6.1.14 Next Steps

That's the end of the walkthrough portion of this guide! You should continue to explore what the game has to offer. There's quite a few features that aren't covered or mentioned in this guide, and even more that get unlocked as you continue to play!

Also, check out the *Netscript* documentation to see what it has to offer. Writing scripts to perform and automate various tasks is where most of the fun in the game comes from (in my opinion)!

The following are a few things you may want to consider doing in the near future.

1.6.1.14.1 Installing Augmentations (and Resetting)

If you've purchased any *Augmentations*, you'll need to install them before you actually gain their effects. Installing `Augmentations` is the game's "soft-reset" or "prestige" mechanic. You can *read more details about it here*.

To install your `Augmentations`, click the `Augmentations` tab on the left-hand navigation menu (*Keyboard shortcut* `Alt + a`). You will see a list of all of the `Augmentations` you have purchased. Below that, you will see a button that says `Install Augmentations`. Be warned, after clicking this there is no way to undo it (unless you load an earlier save).

1.6.1.14.2 Automating the Script Startup Process

Whenever you install `Augmentations`, all of your scripts are killed and you'll have to re-run them. Doing this every time you install `Augmentations` would be very tedious and annoying, so you should write a script to automate the process. Here's a simple example for a startup script. Feel free to adjust it to your liking.

```
// Array of all servers that don't need any ports opened
// to gain root access. These have 16 GB of RAM
var servers0Port = ["sigma-cosmetics",
                    "joesguns",
                    "nectar-net",
                    "hong-fang-tea",
                    "harakiri-sushi"];
```

(continues on next page)

(continued from previous page)

```
// Array of all servers that only need 1 port opened
// to gain root access. These have 32 GB of RAM
var servers1Port = ["neo-net",
                    "zer0",
                    "max-hardware",
                    "iron-gym"];

// Copy our scripts onto each server that requires 0 ports
// to gain root access. Then use nuke() to gain admin access and
// run the scripts.
for (var i = 0; i < servers0Port.length; ++i) {
    var serv = servers0Port[i];

    scp("early-hack-template.script", serv);
    nuke(serv);
    exec("early-hack-template.script", serv, 6);
}

// Wait until we acquire the "BruteSSH.exe" program
while (!fileExists("BruteSSH.exe")) {
    sleep(60000);
}

// Copy our scripts onto each server that requires 1 port
// to gain root access. Then use brutessh() and nuke()
// to gain admin access and run the scripts.
for (var i = 0; i < servers1Port.length; ++i) {
    var serv = servers1Port[i];

    scp("early-hack-template.script", serv);
    brutessh(serv);
    nuke(serv);
    exec("early-hack-template.script", serv, 12);
}
```

1.6.1.15 Random Tips

- Early on in the game, it's better to spend your money on upgrading RAM and purchasing new servers rather than spending it on Augmentations
- The more money available on a server, the more effective the `hack()` and `grow()` Netscript functions will be. This is because both of these functions use percentages rather than flat values. `hack()` steals a percentage of a server's total available money, and `grow()` increases a server's money by X%.
- There is a limit to how much money can exist on a server. This value is different for each server. The `getServerMaxMoney()` function will tell you this maximum value.
- At this stage in the game, your combat stats (strength, defense, etc.) are not nearly as useful as your hacking stat. Do not invest too much time or money into gaining combat stat exp.

1.6.2 What BitNode should I do?

Warning: This page contains spoilers regarding the game's story/plot-line.

After destroying their first *BitNode*, many players wonder which BitNode they should tackle next. This guide hopefully helps answer that question.

1.6.2.1 Overview of each BitNode

1.6.2.1.1 BitNode-1: Source Genesis

Description The first BitNode created by the Enders to imprison the minds of humans. It became the prototype and testing-grounds for all of the BitNodes that followed. This is the first BitNode that you play through. It has no special modifications or mechanics.

Source-File

Max Level 3

This Source-File lets the player start with 32GB of RAM on his/her home computer when entering a new BitNode, and also increases all of the player's multipliers by:

- Level 1: 16%
- Level 2: 24%
- Level 3: 28%

Difficulty The easiest BitNode

1.6.2.1.2 BitNode-2: Rise of the Underworld

Description Organized crime groups quickly filled the void of power left behind from the collapse of Western government in the 2050s. As society and civilization broke down, people quickly succumbed to the innate human impulse of evil and savagery. The organized crime factions quickly rose to the top of the modern world.

In this BitNode:

- Your hacking level is reduced by 20%
- The growth rate and maximum amount of money available on servers are significantly decreased
- The amount of money gained from crimes and Infiltration is tripled
- Certain Factions (Slum Snakes, Tetrads, The Syndicate, The Dark Army, Speakers for the Dead, NiteSec, The Black Hand) give the player the ability to form and manage their own gangs. These gangs will earn the player money and reputation with the corresponding Faction
- Every Augmentation in the game will be available through the Factions listed above
- For every Faction NOT listed above, reputation gains are halved
- You will no longer gain passive reputation with Factions

Source-File

Max Level 3

This Source-File allows you to form gangs in other BitNodes once your karma decreases to a certain value. It also increases the player's crime success rate, crime money, and charisma multipliers by:

- Level 1: 24%
- Level 2: 36%
- Level 3: 42%

Difficulty Fairly easy, as hacking is still very profitable and the costs of various purchases/upgrades is not increased. The gang mechanic may seem strange as its very different from anything else, but it can be very powerful once you get the hang of it.

1.6.2.1.3 BitNode-3: Corporatocracy

Description Our greatest illusion is that a healthy society can revolve around a single-minded pursuit of wealth. Sometime in the early 21st century economic and political globalization turned the world into a corporatocracy, and it never looked back. Now, the privileged elite will happily bankrupt their own countrymen, decimate their own community, and evict their neighbors from houses in their desperate bid to increase their wealth. In this BitNode you can create and manage your own corporation. Running a successful corporation has the potential of generating massive profits. All other forms of income are reduced by 75%. Furthermore:

- The price and reputation cost of all Augmentations is tripled
- The starting and maximum amount of money on servers is reduced by 75%
- Server growth rate is reduced by 80%
- You now only need 75 favour with a faction in order to donate to it, rather than 150

Source-File

Max Level 3

This Source-File lets you create corporations on other BitNodes (although some BitNodes will disable this mechanic). This Source-File also increases your charisma and company salary multipliers by:

- Level 1: 8%
- Level 2: 12%
- Level 3: 14%

Difficulty Somewhat-steep learning curve as you learn how to use and manage Corporations. Afterwards, however, the BitNode is easy as Corporations can be very profitable.

1.6.2.1.4 BitNode-4: The Singularity

Description The Singularity has arrived. The human race is gone, replaced by artificially superintelligent beings that are more machine than man.

In this BitNode, progressing is significantly harder:

- Experience gain rates for all stats are reduced.
- Most methods of earning money will now give significantly less.

In this BitNode you will gain access to a new set of Netscript Functions known as Singularity Functions. These functions allow you to control most aspects of the game through scripts, including working for factions/companies, purchasing/installing Augmentations, and creating programs.

Source-File

Max Level 3

This Source-File lets you access and use the Singularity Functions in other BitNodes. Each level of this Source-File will reduce RAM costs.

Difficulty: Depending on what Source-Files you have unlocked before attempting this BitNode, it can range from easy to moderate.

1.6.2.1.5 BitNode-5: Artificial Intelligence

Description They said it couldn't be done. They said the human brain, along with its consciousness and intelligence, couldn't be replicated. They said the complexity of the brain results from unpredictable, nonlinear interactions that couldn't be modeled by 1's and 0's. They were wrong.

In this BitNode:

- The base security level of servers is doubled
- The starting money on servers is halved, but the maximum money remains the same
- Most methods of earning money now give significantly less
- Infiltration gives 50% more reputation and money
- Corporations have 50% lower valuations and are therefore less profitable
- Augmentations are more expensive
- Hacking experience gain rates are reduced

Source-File

Max Level 3

This Source-File grants you a special new stat called Intelligence.

Intelligence is unique because it is permanent and persistent (it never gets reset back to 1). However gaining Intelligence experience is much slower than other stats, and it is also hidden (you won't know when you gain experience and how much). Higher Intelligence levels will boost your production for many actions in the game.

In addition, this Source-File will unlock the `getBitNodeMultipliers()` Netscript function, and will also raise all of your hacking-related multipliers by:

- Level 1: 8%
- Level 2: 12%
- Level 3: 14%

Difficulty Depending on what Source-Files you have unlocked before attempting this BitNode, it can range from easy to moderate.

1.6.2.1.6 BitNode-6: Bladeburners

Description In the middle of the 21st century, OmniTek Incorporated began designing and manufacturing advanced synthetic androids, or Synthoids for short. They achieved a major technological breakthrough in the sixth generation of their Synthoid design, called MK-VI, by developing a hyperintelligent AI. Many argue that this was the first sentient AI ever created. This resulted in Synthoid models that were stronger, faster, and more intelligent than the humans that had created them.

In this BitNode you will be able to access the Bladeburner Division at the NSA, which provides a new mechanic for progression. Furthermore:

- Hacking and Hacknet Nodes will be less profitable
- Your hacking level is reduced by 65%
- Hacking experience gain from scripts is reduced by 75%
- Corporations have 80% lower valuations and are therefore less profitable
- Working for companies is 50% less profitable
- Crimes and Infiltration are 25% less profitable

Source-File**Max Level 3**

This Source-File allows you to access the NSA's Bladeburner Division in other BitNodes. In addition, this Source-File will raise both the level and experience gain rate of all your combat stats by:

- Level 1: 8%
- Level 2: 12%
- Level 3: 14%

Difficulty Initially difficult due to the fact that hacking is no longer profitable and you have to learn a new mechanic. After you get the hang of the Bladeburner mechanic, however, it becomes moderately easy.

1.6.2.1.7 BitNode-7: Bladeburners 2079

Description In the middle of the 21st century, you were doing cutting-edge work at OmniTek Incorporated as part of the AI design team for advanced synthetic androids, or Synthoids for short. You helped achieve a major technological breakthrough in the sixth generation of the company's Synthoid design, called MK-VI, by developing a hyperintelligent AI. Many argue that this was the first sentient AI ever created. This resulted in Synthoid models that were stronger, faster, and more intelligent than the humans that had created them.

In this BitNode you will be able to access the Bladeburner API, which allows you to access Bladeburner functionality through Netscript. Furthermore:

- The rank you gain from Bladeburner contracts/operations is reduced by 40%
- Bladeburner skills cost twice as many skill points
- Augmentations are 3x more expensive
- Hacking and Hacknet Nodes will be significantly less profitable
- Your hacking level is reduced by 65%
- Hacking experience gain from scripts is reduced by 75%
- Corporations have 80% lower valuations and are therefore less profitable
- Working for companies is 50% less profitable
- Crimes and Infiltration are 25% less profitable

Source-File**Max Level 3**

This Source-File allows you to access the Bladeburner Netscript API in other BitNodes. In addition, this Source-File will increase all of your Bladeburner multipliers by:

- Level 1: 8%

- Level 2: 12%
- Level 3: 14%

Difficulty Slightly more difficult than BitNode-6. However, you will be able to automate more aspects of the Bladeburner feature, which means it will be more passive.

1.6.2.1.8 BitNode-8: Ghost of Wall Street

Description You are trying to make a name for yourself as an up-and-coming hedge fund manager on Wall Street.

In this BitNode:

- You start with \$250 million
- The only way to earn money is by trading on the stock market
- You start with a WSE membership and access to the TIX API
- You are able to short stocks and place different types of orders (limit/stop)
- You can immediately donate to factions to gain reputation

Source-File

Max Level 3

This Source-File grants the following benefits:

- Level 1: Permanent access to WSE and TIX API
- Level 2: Ability to short stocks in other BitNodes
- Level 3: Ability to use limit/stop orders in other BitNodes

This Source-File also increases your hacking growth multipliers by:

- Level 1: 12%
- Level 2: 18%
- Level 3: 21%

Difficulty Very difficult until you unlock the Four Sigma (4S) Market Data API. After you unlock the API however, it becomes moderately easy.

1.6.2.1.9 BitNode-9: Hactocracy

Description When Fulcrum Technologies released their open-source Linux distro Chapeau, it quickly became the OS of choice for the underground hacking community. Chapeau became especially notorious for powering the Hacknet, a global, decentralized network used for nefarious purposes. Fulcrum quickly abandoned the project and dissociated themselves from it.

This BitNode unlocks the Hacknet Server, an upgraded version of the Hacknet Node. Hacknet Servers generate hashes, which can be spent on a variety of different upgrades.

In this BitNode:

- Your stats are significantly decreased
- You cannot purchase additional servers
- Hacking is significantly less profitable

Source-File**Max Level 3**

This Source-File grants the following benefits:

- Level 1: Permanently unlocks the Hacknet Server in other BitNodes
- Level 2: You start with 128GB of RAM on your home computer when entering a new BitNode
- Level 3: Grants a highly-upgraded Hacknet Server when entering a new BitNode

(Note that the Level 3 effect of this Source-File only applies when entering a new BitNode, NOT when installing Augmentations.)

Difficulty Hard

1.6.2.1.10 BitNode-10: Digital Carbon

Description In 2084, VitaLife unveiled to the world the Persona Core, a technology that allowed people to digitize their consciousness. Their consciousness could then be transferred into Synthoids or other bodies by transmitting the digitized data. Human bodies became nothing more than ‘sleeves’ for the human consciousness. Mankind had finally achieved immortality - at least for those that could afford it.

This BitNode unlocks Sleeve technology. Sleeve technology allows you to:

1. Re-sleeve: Purchase and transfer your consciousness into a new body
2. Duplicate Sleeves: Duplicate your consciousness into Synthoids, allowing you to perform different tasks synchronously

In this BitNode:

- Your stats are significantly decreased
- All methods of gaining money are half as profitable (except Stock Market)
- Purchased servers are more expensive, have less max RAM, and a lower maximum limit
- Augmentations are 5x as expensive and require twice as much reputation

Source-File**Max Level 3**

This Source-File unlocks Sleeve technology in other BitNodes. Each level of this Source-File also grants you a Duplicate Sleeve.

Difficulty Hard

1.6.2.1.11 BitNode-11: The Big Crash

Description The 2050s was defined by the massive amounts of violent civil unrest and anarchic rebellion that rose all around the world. It was this period of disorder that eventually lead to the governmental reformation of many global superpowers, most notably the USA and China. But just as the world was slowly beginning to recover from these dark times, financial catastrophe hit. In many countries, the high cost of trying to deal with the civil disorder bankrupted the governments. In all of this chaos and confusion, hackers were able to steal billions of dollars from the world’s largest electronic banks, prompting an international banking crisis as governments were unable to bail out insolvent banks. Now, the world is slowly crumbling in the middle of the biggest economic crisis of all time.

In this BitNode:

- Your hacking stat and experience gain are halved
- The starting and maximum amount of money available on servers is significantly decreased
- The growth rate of servers is significantly reduced
- Weakening a server is twice as effective
- Company wages are decreased by 50%
- Corporation valuations are 99% lower and are therefore significantly less profitable
- Hacknet Node production is significantly decreased
- Crime and Infiltration are more lucrative
- Augmentations are twice as expensive

Source-File

Max Level 3

Destroying this BitNode will give you Source-File 11, or if you already have this Source-File it will upgrade its level up to a maximum of 3. This Source-File makes it so that company favor increases BOTH the player's salary and reputation gain rate at that company by 1% per favor (rather than just the reputation gain). This Source-File also increases the player's company salary and reputation gain multipliers by:

- Level 1: 32%
- Level 2: 48%
- Level 3: 56%

Difficulty Hard

1.6.2.1.12 BitNode-12: The Recursion

Description Every time this BitNode is destroyed, it becomes slightly harder.

Source-File

Max Level Infinity

Each level of Source-File 12 will let you start with Neuroflux Governor equal to the level of this Source-File.

This BitNode is meant to be done passively or when waiting for new content.

Difficulty Initially very easy, but then it (obviously) becomes harder as you continue to do it.

1.6.2.2 Recommended BitNodes

As a player, you are not forced to tackle the BitNodes in any particular order. You are free to choose whichever ones you want. The “best” order can vary between players, depending on what you like to do any what kind of player you are. In general, here are the recommended BitNodes for different things:

1.6.2.2.1 For fast progression

Note: This does not recommend the absolute fastest path, as I don't know what exactly the fastest path is. But it does recommend the BitNodes that are commonly considered to be optimal by players.

1. Repeat **BitNode-1: Source Genesis** until you max out its Source-File. Its Source-File is extremely powerful, as it raises all multipliers by a significant amount.
2. Do **BitNode-5: Artificial Intelligence** once or twice. The intelligence stat it unlocks will gradually build up as you continue to play the game, and will be helpful in the future. The Source-File also provides hacking multipliers, which are strong because hacking is typically one of the best ways of earning money.
3. (Optional) Consider doing **BitNode-4: The Singularity**. Its Source-File does not directly make you more powerful in any way, but it does unlock the [Singularity API](#) which let you automate significantly more aspects of the game.
4. Do **BitNode-3: Corporatocracy** once to unlock the Corporation mechanic. This mechanic has high profit potential.
5. Do **BitNode-6: Bladeburners** once to unlock the Bladeburners mechanic. The Bladeburner mechanic is useful for some of the future BitNodes (such as 9 and 10).
6. Do **BitNode-9: Hacktocracy** to unlock the Hacknet Server mechanic. You can consider repeating it as well, as its Level 2 and 3 effects are pretty helpful as well.

Todo: To be continued as more BitNodes get added

1.6.2.2.2 For the strongest Source-Files

Note that the strongest Source-Files are typically rewarded by the hardest BitNodes.

The strongest Source-File is that from **BitNode-1: Source Genesis**, as it raises all multipliers by a significant amount.

Similarly, the Source-File from **BitNode-12: The Recursion** is also very strong because it raises all multipliers. Each level of Source-File 12 is fairly weak, but its effectiveness gets better over time since the effects of Source-Files and Augmentations are multiplicative with each other.

The Source-File from **BitNode-9: Hacktocracy** is good because it unlocks the Hacknet Server mechanic. The Hacknet Server mechanic causes Hacknet Nodes to produce a new currency called *hashes*, rather than money. *Hashes* can be spent on powerful upgrades that benefit your hacking, Corporation, Bladeburner, etc.

The Duplicate Sleeves granted by the Source-File from **BitNode-10: Digital Carbon** are strong, but only after you have several of them and have spent some time/money upgrading them.

1.6.2.2.3 For more scripting/hacking

BitNode-4: The Singularity unlocks the [Singularity API](#), which can be used to automate many different aspects of the game, including working for factions/companies, purchasing & installing Augmentations, and creating programs

BitNode-6 and **BitNode-7** unlock Bladeburner and its corresponding [Netscript API](#). This allows you to automate an entire new mechanic.

BitNode-2: Rise of the Underworld also unlocks a new mechanic and Netscript API for automating it (the Gang mechanic). However, it is not as interesting as Bladeburner (in my opinion)

BitNode-9: Hacktocracy unlocks the Hacknet Server mechanic and several new functions in the [Hacknet Node API](#) for using it.

1.6.2.2.4 For new mechanics

BitNode-2: Rise of the Underworld unlocks a new mechanic in which you can manage a gang. Gangs earn you money and can be very profitable once they get large and powerful. The biggest benefit of gangs, however, is that they make all Augmentations available to you through their corresponding faction.

BitNode-3: Corporatocracy unlocks a new mechanic in which you can manage a corporation. You can earn money through Corporations by selling your stocks, or by configuring your corporation to pay dividends to shareholders. If your Corporation gets big enough, it can also bribe factions in exchange for faction reputation.

BitNode-6: Bladeburners unlocks a new mechanic that centers around combat rather than hacking. The main benefit of the Bladeburner mechanic is that it offers a new method of destroying a BitNode.

BitNode-9: Hacktocracy unlocks the Hacknet Server, which is an upgraded version of a Hacknet Node. The Hacknet Server generates a computational unit called a *hash*. *Hashes* can be spent on a variety of different upgrades that can benefit your hacking, Corporation, Bladeburner progress, and more. It transforms the Hacknet Node from a simple money-generator to a more interesting mechanic.

BitNode-10: Digital Carbon unlocks two new mechanics: Re-Sleeving and Duplicate Sleeves.

1.6.2.2.5 For a Challenge

In general, the higher BitNodes are more difficult than the lower ones. **BitNode-12: The Recursion** is an obvious exception as it gets progressively harder.

BitNode-8: Ghost of Wall Street provides a unique challenge as the only method of earning money in that BitNode is through trading at the stock market.

1.7 Tools & Resources

1.7.1 Official Script Repository

There are plans to create an official repository of Bitburner scripts. As of right now, this is not a priority and has not been started. However, if you'd like to contribute scripts now, you can find the repository [here](#) and submit pull requests.

1.7.2 Visual Studio Code Extension

One user created a Bitburner extension for the Visual Studio Code (VSCode) editor.

This extension includes several features such as:

- Dynamic RAM calculation
- RAM Usage breakdown
- Typescript definition files with jsdoc comments
- Netscript syntax highlighting

You can find more information and download links [on this reddit post](#).

1.8 Changelog

1.8.1 v1.6.3 - 2022-04-01 Few stanek fixes

Stanek Gift

- Has a minimum size of 2x3
- Active Fragment property 'avgCharge' renamed to 'highestCharge'
- Formula for fragment effect updated to make 561% more sense. Now you can charge to your heart content.
- Logs for the 'chargeFragment' function updated.

Misc.

- Nerf noodle bar.

1.8.2 v1.6.0 - 2022-03-29 Grafting

** Vitalife secret lab **

- A new mechanic called Augmentation Grafting has been added. Resleeving has been removed.
- Credit to @nickofolas for his incredible work.

** Stanek **

- **BREAKING:** Many functions in the stanek API were renamed in order to avoid name collision with things like Map.prototype.get

** UI **

- Major update to Sleeve, Gang UI, and Create Program (@nickofolas)
- re-add pre tags to support slash n in prompt (@jacktose)
- Tabelize linked output of 'ls' (@Master-Guy)
- Add the ability to filter open scripts (@physical)
- Add minHeight to editor tabs (@nickofolas)
- Properly expand gang equipment cards to fill entire screen (@nickofolas)
- Add shortcut to Faction augmentations page from FactionsRoot (@nickofolas)
- Fix extra space on editor tabs (@nickofolas)
- Present offline message as list (@DSteve595)
- add box showing remaining augments per faction (@jjayeon)
- Add tab switching support to vim mode (@JParisFerrer)
- Show current task on gang management screen (@zeddrak)
- Fix for ui of gang members current task when set via api (@physical)
- Don't hide irrelevant materials if their stock is not empty and hide irrelevant divisions from Export (@SagePtr)
- Fix regex to enable alpha transparency hex codes (8 digits) (@surdaft)

** API **

- Added dark web functions to ns api

- BREAKING: purchaseTor() should returns true if player already has Tor. (@DavidGrinberg, @waffleattack)
- Implement getBonusTime in Corporation API (@t-wolfeadam)
- Added functions to purchase TIX and WSI (@incubusnb)
- purchaseSleeveAug checks shock value (@incubusnb)
- Fix bug with hacknet api
- Fix spendHashes bug
- Added 0 cost of asleep() (@Master-Guy)
- Fix some misleading corporation errors (@TheRealMaxion)
- expose the inBladeburner on the player object (@physical)
- added ram charge for stanek width and height (@physical)
- Fix sufficient player money check to buy back shares. (@ChrissiQ)
- Fix Static Ram Circumventing for some NS functions (@CrafterKolyan)
- added CorporationSoftCap to NetscriptDefinitions (@physical)
- Added definition of autocomplete() 'data' argument. (@tigercat2000)
- Adding support for text/select options in Prompt command (@PhilipArmstead)
- Added the ability to exportGame via api (@physical)

**** Arcade ****

- Added an arcade to New Tokyo where you can play a 4 year old version of bitburner.

**** Misc. ****

- Add a warning triggered while auto-saves are off. (@MartinFournier)
- Log info for field analysis now displays actual rank gained. (@ApamNapat)
- Removed BladeburnerSkillCost from skill point cost description. (@ApamNapat)
- Fix handling for UpArrow in bladeburner console. (@dowinter)
- Add GitHub action to check PRs for generated files. (@MartinFournier)
- Cap Staneks gift at 25x25 to prevent crashes. (@waffleattack)
- Remove old & unused files from repository. (@MartinFournier)
- Factions on the factions screens are sorted by story progress / type. (@physical)
- Fix log manager not picking up new runs of scripts. (@physical)
- Added prettier to cicd.
- UI improvements (@physical)
- Documentation / Typos (@nanogyth, @Master-Guy, @incubusnb, @ApamNapat, @physical, @SagePtr)
- Give player code a copy of Division.upgrades instead of the live object (@Ornedan)
- Fix bug with small town achievement.
- Fix bug with purchaseSleeveAug (@physical)
- Check before unlocking corp upgrade (@gianfun)
- General codebase improvements. (@physical, @Master-Guy, @ApamNapat)

- Waiting on promises in NS1 no longer freezes the script. (@Master-Guy)
- Fix bug with missing ramcost for tFormat (@TheMas3212)
- Fix crash with new prompt
- Quick fix to prevent division by 0 in terminal (@Master-Guy)
- removed ip references (@physical, @Master-Guy)
- Terminal now supports 'ls -l'
- Fix negative number formatting (@Master-Guy)
- Fix unique ip generation (@InDieTasten)
- remove terminal command theme from docs (@physical)
- Fix 'Augmentations Left' with gang factions (@nickofolas)
- Attempt to fix 'bladeburner.process()' early routing issue (@MartinFournier)
- work in progress augment fix (@physical)
- Fixes missing space in Smart Supply (@TheRealMaxion)
- Change license to Apache 2 with Commons Clause
- updated regex sanitization (@mbrannen)
- Sleeve fix for when faction isnt found (@physical)
- Fix editor "close" naming (@physical)
- Fix bug with sleeves where some factions would be listed as workable. (@physical)
- Fix research tree of product industries post-prestige (@pd)
- Added a check for exisiting industry type before expanding (@physical)
- fix hackAnalyzeThreads returning infinity (@chrisrabe)
- Make growthAnalyze more accurate (@dwRch yngqxs)
- Add 'Zoom -> Reset Zoom' command to Steam (@smolgumball)
- Add hasOwnProperty check to GetServer (@SagePtr)
- Speed up employee productivity calculation (@pd)
- Field Work and Security Work benefit from 'share' (@SagePtr)
- Nerf noodle bar.

1.8.3 v1.5.0 - Steam Cloud integration

**** Steam Cloud Saving ****

- Added support for steam cloud saving (@MartinFournier)

**** UI ****

- background now matches game primary color (@nickofolas)
- page title contains version (@MartinFourier)
- Major text editor improvements (@nickofolas)
- Display bonus time on sleeve page (@MartinFourier)

- Several UI improvements (@nickofolas, @smolgumball, @DrCuriosity, @phyzical)
- Fix aug display in alpha (@Dominik Winter)
- Fix display of corporation product equation (@SagePtr)
- Make Bitverse more accessible (@ChrissiQ)
- Make corporation warehouse more accessible (@ChrissiQ)
- Make tab style more consistent (@nickofolas)

** Netscript **

- Fix bug with async.
- Add 'printf' ns function (@Ninetailed)
- Remove blob caching.
- Fix formulas access check (@Ornedan)
- Fix bug in exp calculation (@qcorradi)
- Fix NaN comparison (@qcorradi)
- Fix travelToCity with bad argument (@SlyCedix)
- Fix bug where augs could not be purchased via sing (@reacocard)
- Fix rounding error in donateToFaction (@Risenafis)
- Fix bug with weakenAnalyze (@rhobes)
- Prevent exploit with atExit (@Ornedan)
- Double 'share' power

** Corporations **

- Fix bugs with corp API (@pialot)
- Add smart supply func to corp API (@pd)

** Misc. **

- The file API now allows GET and DELETE (@lordducky)
- Force achievement calculation on BN completion (@SagePtr)
- Cleanup in repository (@MartinFourier)
- Several improvements to the electron version (@MartinFourier)
- Fix bug with casino roulette (@jamie-mac)
- Terminal history persists in savefile (@MartinFourier)
- Fix tests (@jamie-mac)
- Fix crash with electron windows tracker (@smolgumball)
- Fix BN6/7 passive reputation gain (@BrianLDev)
- Fix Sleeve not resetting on install (@waffleattack)
- Sort joined factions (@jjayeon)

- Update documentation / typo (@lethern, @Meowdoleon, @JohnnyUrosevic, @JosephDavidTalbot, @pd, @lethern, @lordducky, @zeddrak, @fearnlj01, @reasonablytall, @MatthewTh0, @SagePtr, @man-niL, @Jedimaster4559, @loganville, @Arrow2thekn33, @wdpk, @fwolfst, @fschoenfeldt, @Waladil, @AdamTReineke, @citrusmunch, @factubsio, @ashtongreen, @ChrissiQ, @DJ-Laser, @waffleattack, @ApamNapat, @CrafterKolyan, @DSteve595)
- Nerf noodle bar.

1.8.4 v1.4.0 - 2022-01-18 Sharing is caring

** Computer sharing **

- A new mechanic has been added, it's is invoked by calling the new function 'share'. This mechanic helps you farm reputation faster.

** gang **

- Installing augs means losing a little bit of ascension multipliers.

** Misc. **

- Prevent gang API from performing actions for the type of gang they are not. (@TheMas3212)
- Fix donation to gang faction. (@TheMas3212)
- Fix gang check crashing the game. (@TheMas3212)
- Make time compression more robust.
- Fix bug with scp.
- Add zoom to steam version. (@MartinFourier)
- Fix donateToFaction accepts donation of NaN. (@woody-lam-cwl)
- Show correct hash capacity gain on cache level upgrade tooltip. (@woody-lam-cwl)
- Fix tests (@woody-lam-cwl)
- Fix cache tooltip (@woody-lam-cwl)
- Added script to prettify save file for debugging (@MartinFourier)
- Update documentation / typos (@theit8514, @thadguidry, @tigerCat2000, @SlyCedix, @Spacejoker, @KenJohansson, @Ornedan, @JustAnOkapi, @nickofolas, @philmarmstead, @TheMas3212, @dcragusa, @XxKingsX-Pinu, @paiv, @smolgumball, @zeddrak, @stinky-lizard, @nickofolas, @Feodoric, @daanflore, @markusariliu, @mstruebing, @erplsf, @waffleattack, @Dexalt142, @AIT-OLPE, @deathly809, @BuckA-Mayzing, @MartinFourier, @pigalot, @lethern)
- Fix BN3+ achievement (@SagePtr)
- Fix reputation carry over bug (@TheMas3212)
- Add button to exit infiltrations (@TheMas3212)
- Add dev menu achievement check (@TheMas3212)
- Add 'host' config for electron server (@MartinFourier)
- Suppress save toast only works for autosave (@MartinFourier)
- Fix some achievements not triggering with 'backdoor' (@SagePtr)
- Update Neuroflux Governor description.
- Fix bug with electron server.

- Fix bug with corporation employee assignment function (@Ornedan)
- Add detailed information to terminal 'mem' command (@MartinFourier)
- Add savestamp to savefile (@MartinFourier)
- Dev menu can apply export bonus (@MartinFourier)
- Icarus message no longer applies on top of itself (@Feodoric)
- purchase augment via API can no longer buy Neuroflux when it shouldn't (@Feodoric)
- Syntax highlighter should be smarter (@neuralsim)
- Fix some miscalculation when calculating money stolen (@zeddrak)
- Fix max cache achievement working with 0 cache (@MartinFourier)
- Add achievements in the game, not just steam (@MartinFourier)
- Overflow hash converts to money automatically (@MartinFourier)
- Make mathjax load locally (@MartinFourier)
- Make favor calculation more efficient (@kittycat2002)
- Fix some scripts crashing the game on startup (@MartinFourier)
- Toasts will appear above tail window (@MartinFourier)
- Fix issue that can cause terminal actions to start on one server and end on another (@MartinFourier)
- Fix 'fileExists' not correctly matching file names (@TheMas3212)
- Refactor some code to be more efficient (@TheMas3212)
- Fix exp gain for terminal grow and weaken (@nickofolas)
- Refactor script death code to reject waiting promises instead of resolving (@Ornedan)
- HP recalculates on defense exp gain (@TheMas3212)
- Fix log for ascendMember (@TheMas3212)
- Netscript ports clear on reset (@TheMas3212)
- Fix bug related to company (@TheMas3212)
- Fix bug where corporation handbook would not be correctly added (@TheMas3212)
- Servers in hash upgrades are sorted alpha (@MartinFourier)
- Fix very old save not properly migrating augmentation renamed in 0.56 (@MartinFourier)
- Add font height and line height in theme settings (@MartinFourier)
- Fix crash when quitting job (@MartinFourier)
- Added save file validation system (@TheMas3212)
- React and ReactDOM are now global objects (@pigalot)
- 'nano' supports globs (@smolgumball)
- Character overview can be dragged (@MartinFourier)
- Job page updates in real time (@nickofolas)
- Company favor gain uses the same calculation as faction, this is just performance the value didn't change (@nickofolas)

- ns2 files work with more import options (@theit8514)
- Allow autocomplete for partial executables (@nickofolas)
- Add support for contract completion (@nickofolas)
- 'ls' link are clickable (@smolgumball)
- Prevent steam from opening external LOCAL files (@MartinFourier)
- Fix a bug with autocomplete (@Feodoric)
- Optimise achievement checks (@Feodoric)
- Hacknet server achievements grant associated hacknet node achievement (@Feodoric)
- Fix display bug with hacknet (@Feodoric)
- 'analyze' now says if the server is backdoored (@deathly809)
- Add option to exclude running script from save (@MartinFourier)
- Game now catches more errors and redirects to recovery page (@MartinFourier)
- Fix bug with autocomplete (@nickofolas)
- Add tooltip to unfocus work (@nickofolas)
- Add detailst overview (@MartinFourier)
- Fix focus bug (@deathly809)
- Fix some NaN handling (@deathly809)
- Added 'mv' ns function (@deathly809)
- Add focus argument to some singularity functions (@nickofolas)
- Fix some functions not disabling log correctly (@deathly809)
- General UI improvements (@nickofolas)
- Handle steamworks errors gracefully (@MartinFourier)
- Fix some react component not unmounting correctly (@MartinFourier)
- 'help' autocompletes (@nickofolas)
- No longer push all achievements to steam (@Ornedan)
- Recovery page has more information (@MartinFourier)
- Added 'getGameInfo' ns function (@MartinFourier)
- SF3.3 unlocks all corp API (@pigalot)
- Major improvements to corp API (@pigalot)
- Prevent seed money outside BN3 (@pigalot)
- Fix bug where using keyboard shortcuts would crash if the feature is not available (@MartinFourier)
- Sidebar remains opened/closed on save (@MartinFourier)
- Added tooltip to sidebar when closed (@MartinFourier)
- Fix bug where Formulas.exe is not available when starting BN5 (@TheMas3212)
- Fix CI (@tvanderpol)
- Change shortcuts to match sidebar (@MartinFourier)

- Format gang respect (@attrib)
- Add modal to text editor with ram details (@nickofolas)
- Fix several bugs with singularity focus (@nickofolas)
- Nerf noodle bar.

1.8.5 v1.3.0 - 2022-01-04 Cleaning up

** External IDE integration **

- The Steam version has a webserver that allows integration with external IDEs. A VSCode extension is available on the market place. (The documentation for the ext. isn't written yet)

** Source-Files **

- SF4 has been reworked.
- New SF -1.

** UI **

- Fix some edge case with skill bat tooltips (@MartinFournier)
- Made some background match theme color (@Kejikus)
- Fix problem with script editor height not adjusting correctly (@billyvg)
- Fix some formatting issues with Bladeburner (@MartinFournier, @nickofolas)
- Fix some functions like 'alert' format messages better (@MageKing17)
- Many community themes added.
- New script editor theme (@Hedrauta, @Dexalt142)
- Improvements to tail windows (@theit8514)
- Training is more consise (@mikomyazaki)
- Fix Investopedia not displaying properly (@JotaroS)
- Remove alpha from theme editor (@MartinFournier)
- Fix corporation tooltip not displaying properly (@MartinFournier)
- Add tooltip on backdoored location names (@MartinFournier)
- Allow toasts to be dismissed by clicking them (@nickofolas)
- Darkweb item listing now shows what you own. (@hexnaught)

** Bug fix **

- Fix unit tests (@MartinFournier)
- Fixed issue with 'cat' and 'read' not finding foldered files (@Nick-Colclasure)
- Buying on the dark web will remove incomplete exe (@hexnaught)
- Fix bug that would cause the game to crash trying to go to a job without a job (@hexnaught)
- purchaseServer validation (@nickofolas)
- Script Editor focuses code when changing tab (@MartinFournier)
- Fix script editor for .txt files (@65-7a)

- Fix 'buy' command not displaying correctly. (@hexnaught)
- Fix hackAnalyzeThread returning NaN (@mikomyazaki)
- Electron handles exceptions better (@MageKing17)
- Electron will handle 'unresponsive' event and present the opportunity to reload the game with no scripts (@MartinFournier)
- Fix 'cp' between folders (@theit8514)
- Fix throwing null/undefined errors (@nickofolas)
- Allow shortcuts to work when unfocused (@MageKing17)
- Fix some dependency issue (@locriacyber)
- Fix corporation state returning an object instead of a string (@antonvmironov)
- Fix 'mv' overwriting files (@theit8514)
- Fix joesguns not being influenced by hack/grow (@dou867, @MartinFournier)
- Added warning when opening external links. (@MartinFournier)
- Prevent applying for positions that aren't offered (@TheMas3212)
- Import has validation (@MartinFournier)

**** Misc. ****

- Added vim mode to script editor (@billyvg)
- Clean up script editor code (@Rez855)
- 'cat' works on scripts (@65-7a)
- Add wordWrap for Monaco (@MartinFournier)
- Include map bundles in electron for easier debugging (@MartinFournier)
- Fix importing very large files (@MartinFournier)
- Cache program blob, reducing ram usage of the game (@theit8514)
- Dev menu can set server to \$0 (@mikomyazaki)
- 'backdoor' allows direct connect (@mikomyazaki)
- Github workflow work (@MartinFournier)
- workForFaction / workForCompany have a new parameter (@theit8514)
- Alias accept single quotes (@sporkwitch, @FaintSpeaker)
- Add grep options to 'ps' (@maxtimum)
- Added buy all option to 'buy' (@anthonydroberts)
- Added more shortcuts to terminal input (@Frank-py)
- Refactor some port code (@ErzengelLichtes)
- Settings to control GiB vs GB (@ErzengelLichtes)
- Add electron option to export save game (@MartinFournier)
- Electron improvements (@MartinFournier)
- Expose some notifications functions to electron (@MartinFournier)

- Documentation (@MartinFournier, @cyn, @millenniumAmbiguity, @2PacIsAlive, @TheCoderJT, @hex-naught, @sschmidTU, @FOLLGAD, @Hedrauta, @Xynrati, @mikomyazaki, @Icehawk78, @aaronransley, @TheMas3212, @Hedrauta, @alkemann, @ReeseJones, @amclark42, @thadguidry, @jasonhaxstuff, @pan-kuleczka, @jhollowe, @ApatheticsAnonymous, @erplsf, @daanflore, @nickofolas, @Kebap, @smolgumball, @woody-lam-cwl)

1.8.6 v1.1.0 - 2021-12-18 You guys are awesome (community because they're god damn awesome)

** Script Editor **

- The text editor can open several files at once. (@Rez855 / @Shadow72) It's not perfect so keep the feedback coming.

** Steam **

- Windows has a new launch option that lets player start with killing all their scripts This is a safety net in case all the other safety nets fail.
- Linux has several launch options that use different flags for different OS.
- Debug and Fullscreen are available in the window utility bar.
- Tried (and maybe failed) to make the game completely kill itself after closing. This one I still don't know wtf is going.
- No longer has background throttling.
- Default color should be pitch black when loading
- Add BN13: Challenge achievement.

** Tutorial **

- I watched someone play bitburner on youtube and reworked part of the tutorial to try to make some parts of the game clearer. https://www.youtube.com/watch?v=-_JETXff4Zo
- Add option to restart tutorial.

** Netscript **

- getGangInformation returns more information.
- getAscensionResult added
- getMemberInformation returns more info
- Formulas API has new functions for gang.
- Added documentation for corp API.
- exec has clearer error message when you send invalid data.
- getServer returns all defined field for hacknet servers.
- Fix a bug with scp multiple files (@theit8514)
- Stack traces should be smarter at replacing blobs with filenames
- Fix a weird error message that would occur when throwing raw strings.
- Fix shortcuts not working.
- Re-added setFocus and isFocused (@theit8514)
- new function getHashUpgrades (@MartinFournier)

- enableLog accepts “ALL” like disableLog (@wynro)
- toast() doesn't crash on invalid data (@ivanjermakov)
- alert() doesn't crash on invalid data (@Siern)
- Fixed an issue where scripts don't run where they should.
- Sleeve getInformation now returns cha
- getServer does work with no argument now
- workForFaction returns false when it mistakenly returned null

** Character Overview **

- The character overview now shows the amount of exp needed to next level (@MartinFournier)

** Misc. **

- Add option to suppress Game Saved! toasts (@MartinFournier)
- Fix bug where ctrl+alt+j was eaten by the wrong process. (@billyvg)
- Theme Editor lets you paste colors (@MartinFournier)
- ctrl + u/k/w should work on terminal (@billyvg)
- Game now shows commit number, this is mostly for me. (@MartinFourier)
- running a bad script will give a clearer error message (@TheCoderJT)
- Default terminal capacity is maximum (@SayntGarmo)
- Fix problems with cp and mv (@theit8514)
- Make monaco load fully offline for players behind firewalls.
- change beginner guide to use n00dles instead of foodnstuff
- BN13 is harder
- nerf int gain from manualHack
- Fix UI displaying wrong stats (@DJMatch3000)
- Fix button not disabling as it should.
- New location in Ishima.
- Add setting to suppress stock market popups.
- Typo fixes (@Hedrauta, @cvr-119, @Ationi, @millennIumAMbiguity @TealKoi, @TheCoderJT, @cblte, @2PacIsAlive, @MageKing17, @Xynrati, @Adraxas, @pobiega)
- Fix 100% territory achievement.
- Reword message on active scripts page.
- Fix terminal not clearing after BN
- Remove references to .fconf
- Augmentation pages shows BN difficulty with SF5
- Fix scripts saving on wrong server while 'connect'ing
- Fix gym discount not working.
- Fix scan-analyze not working with timestamps

- Hash upgrades remember last choice.
- Save files now sort by date
- The covenant no longer supports negative memory purchases
- Fix corp shares buyback triggering by pressing enter
- Staneks gift display avg / num charges
- Infiltration rewards no longer decay with better stats
- terminal 'true' is parsed as boolean not string
- tail and kill use autocomplete()
- Fix focus for coding contract
- massive boost to noodle bar.

**** Special Thanks ****

- Special thank you to everyone on Discord who can answer new player questions so I can focus on more important things.

1.8.7 v1.1.0 - 2021-12-03 BN13: They're Lunatics (hydroflame & community)

**** BN13: They're Lunatics ****

- BN13 added.

**** Steam ****

- Tested on all 3 major OS.
- 94 achievements added
- Release is 2021-12-10.

**** Corporation API ****

- Added corporation API. (Unstable)

**** Netscript ****

- tprintf crashes when not giving a format as first arg.
- tprintf no longer prints filename (@BartKoppelmans)
- TIX buy/sell/sellShort all return askprice/bidprice (@Insight)
- getRunningScript now works.
- Fix disableLog for gang and TIX API
- getOwnedSourceFiles is not singularity anymore (makes it easier to share scripts.) (@theit8514)
- true/false is a valid value to send to other scripts.
- workForFaction no longer returns null when trying to work for gang.
- Scripts logging no longer generates the string if logging is disabled. This should give performance boost for some scripts.

**** Gang ****

- Gang with 0 territory can no longer fight
- Territory now caps at exactly 0 or 1.

**** Misc. ****

- Clicking “previous” on the browser will not pretend you had unsaved information allowing you to cancel if needs be.
- Fixed some tail box coloring issue.
- Fixed BladeBurner getCityCommunities ram cost
- The download terminal command no longer duplicate extensions (@Insight)
- Fix #000 on #000 text in blackjack. (@Insight)
- Remove reference to .fconf
- Tail boxes all die on soft reset.
- Fix codign contract focus bug.
- Megacorp factions simply re-invite you instead of auto added on reset. (@theit8514)
- Tail window is bound to html body.
- Infiltration reward is tied to your potential stats, not your actual stats So you won’t lose reward for doing the same thing over and over.
- intelligence lowers program creation requirements.
- Terminal parses true as the boolean, not the string.
- Tail and kill autocomplete using the ns2 autocomplete feature.
- scan-analyze doesn’t take up as many terminal entries.
- GangOtherInfo documentation now renders correctly.
- ActiveScripts search box also searches for script names.
- Infinite money no longer allows for infinite hacknet server.
- Blackjack doesn’t make you lose money twice.
- Recent Scripts is now from most to least recent.
- Fix mathjax ascii art bug in NiteSec.
- Remove warning that the theme editor is slow, it’s only slow in dev mode.
- In BN8 is it possible to reduce the money on a server without gaining any.
- In the options, the timestamp feature has a placeholder explaining the expected format.
- Bunch of doc typo fix. (hydroflame & @BartKoppelmans & @cvr-119)
- nerf noodle bar

1.8.8 v1.0.2 - 2021-11-17 It’s the little things (hydroflame)

**** Breaking (very small I promise!) ****

- buy / sell now return getAskPrice / getBidPrice instead of just price. This should help solve some inconsistencies.

**** Misc. ****

- scripts logs are colorized. Start your log with SUCCESS, ERROR, FAIL, WARN, INFO.
- documentation for scp not say string | string[]

- Donation link updated.
- nerf noodle bar

1.8.9 v1.0.1 - 2021-11-17 New documentation (hydroflame)

** Documentation **

- The new documentation for the netscript API is available at <https://github.com/danielyxie/bitburner/blob/dev/markdown/bitburner.ns.md> This documentation is used in-game to validate the code, in-editor to autocomplete, and for users to reference. This is a huge quality of life improvements for me.

** Reputation **

- Fixed favor not affecting faction work reputation gain (Yeah, I know right?)

** Hacknet **

- Servers are now considered “purchasedByPlayers”

** Script Editor **

- solarized themes now work.

** Corporation **

- Dividends are now much more taxed.
- The 2 upgrades that reduced taxes are now much stronger.

** Misc. **

- Starting / Stopping scripts on hashnet servers immediately updates their hash rate (instead of on the next tick)
- Hacknet has tooltip showing what the result of the upgrade would be.
- Augmentations page displays current price multiplier as well as explains the mechanic.
- Terminal now is 25x stronger.
- Tail boxes use pre-wrap for it's lines.
- Tail boxes allow you to rerun dead scripts.
- Tail boxes can no longer open the same one twice.
- Terminal now autocompletes through aliases.
- Make alter reality harder.
- Fix bladeburner cancelling actions when manually starting anything with Simulacrum.
- Buying hash upgrade to increase uni class or gym training will apply to current class.
- Internally the game no longer uses the decimal library.
- Fix an issue where ‘download *’ would generate weird windows files.
- Timestamps can be set to any format in the options.
- Fix typo in documentation share popup.
- Remove bunch of debug log.
- Fix typo in corporation handbook literature.
- Fix typo in documentation
- Fix duplicate SF -1 exploit. (Yeah, an exploit of exploits, now were meta)

- Fix offline hacking earning being attributed to hacknet.
- nerf noodle bar

1.8.10 v1.0.0 - 2021-11-10 Breaking the API :((blame hydroflame)

** Announcement **

- Several API breaks have been implemented.
- See the v1.0.0 migration guide https://bitburner.readthedocs.io/en/latest/v1.0.0_migration.html
- Everyone gets 10 free neuroflux level.

** Netscript **

- Fix a bug that would cause RAM to not get recalculated.
- New function: `hackAnalyzeSecurity`
- New function: `growthAnalyzeSecurity`
- New function: `weakenAnalyze`

** Script Editor **

- Sometimes warn you about unawaited infinite loops.
- `ns1` functions are now correctly colors in Monokai.

** Programs **

- `Formulas.exe` is a new program that lets you use the formulas API.

** Corporations **

- Real Estate takes up a tiny bit of room.
- Dividends are now taxes exponentially in certain bitnodes.
- UI displays how many level of each corporation upgrade.
- Fix exploit with going public.
- Employee salary no longer increase.

** Documentation **

- The documentation is now autogenerated into `.md` files. It is usable but not yet linked to readthedocs. It's on github.

** Misc. **

- Favor is not internal floating point. Meaning I don't have to save an extra variable.
- Manually starting a Bladeburner action cancels unfocused action.
- Updated description of gang territory to be clearer.
- Hacknet expenses and profit are in different categories.
- Fixed favor equation.
- Toast messages aren't hidden behind work in progress screen.
- Fix bug that made infiltration checkmark look off by one.
- Fix some inconsistency with running files that start or don't start with /

- Can't tail the same window twice.
- Added recovery mode. Hopefully no one will ever have to use it.
- Fix readthedocs
- Programs now give int exp based on time not program.
- Many sing. functions now give int exp.
- Active Scripts page now displays some arguments next to script name.
- Fixed some invisible black text.
- Button colors can be edited.
- Added 2 new colors in the theme editor: background primary and background secondary.
- infiltration uses key instead of keycode so it should work better on non-american keyboards.
- buff noodle bar.

1.8.11 v0.58.0 - 2021-10-27 Road to Steam (hydroflame & community)

**** Announcement ****

- To prepare for Steam we will fix some inconsistencies in the Netscript API. Ideally we can also write a save file migration that will automatically convert all breaking changes in your scripts without any player input.

**** BREAKING (kindof) ****

- All stock market functions are now under the 'stock' namespace, like 'hacknet' However when you load your game with v0.58.0 for the first time it should automatically convert everything.

**** SF -1 ****

- new SF -1: Reality Alteration

**** Gang ****

- Ascension formula now better
- Karma requirement now much lower in most nodes
- Territory heavily penalizes gains
- T.R.P. not available outside BN2.

**** Netscript ****

- It is no longer possible to send anything but strings or numbers to other scripts. (prevents exploits)
- Improve code for some netscript functions (@omuretsu)

**** Script Editor ****

- Added Solarized light/dark as theme (@CalvinTrops)
- Fixed sleeve namespace smart autocomplete.

**** Hacknet Servers ****

- Cores affect grow/weaken like they do on home computer

**** Infiltration ****

- Slash game modified to be easier.

**** Misc. ****

- Fix typo in corp (@Saynt_Garmo)
- Fixed a bug where corp wouldn't let you buyback shares. (@Saynt_Garmo)
- Fixed a bug where sleeves couldn't perform some crimes. (@Saynt_Garmo)
- Hospitalization and Eating noodles are now toasts (@Saynt_Garmo)
- Fixed some repeated code (@omuretsu)
- Fixed Character Overview preventing clicks underneath it even when hidden. (@omuretsu)
- Fixed typo in tutorial. (@omuretsu)
- Create Programs and Factions invitation badges now dissappear when you open their respective pages.
- Add killall script in character overview.
- Fixed bug in corp that made last city production be the production for all cities for newly created product.
- Fix bug that allowed reputation to transfer to new jobs.
- Fixed memory leak with ns2.
- nerf noodle bar

1.8.12 v0.57.0 - 2021-10-16 It was too cheap! (hydroflame & community)

** BREAKING (kindof) **

- purchased server cost now scales exponentially past 2^{10} . I'm going to actually explain this one: Currently the cost of a 2^{20} GB server is 57b Most players can get that before their first install. In an effort to nerf good players a softcap was added. This softcap is different for every BN.

** Script Editor **

- Added a theme that is close to monokai. Unfortunately a full monokai is impossible because Monaco doesn't have a very good tokenizer.
- Opening a file and connecting to a new server will still save the file on the server that the file was opened.

** Netscript **

- New function: alert, which creates a textbox.
- New function: toast, creates a notification in the bottom right.
- New function: upgradeHomeCores (@Saynt_Garmo)
- New function: atExit, allows you to set a callback for when the script closes.
- New kindof function: autocomplete, this allows you to tell the game what it should autocomplete on the terminal.

** Augmentation **

- ENM Core (the Augmentation from The Black Hand with the highest rep cost) rep cost reduced from 250 to 175. This will help new players transition from TBH to BitRunners more easily.

** Bladeburner **

- New general action: Incite Violence. This action adds other action counts but increases chaos.

** Misc. **

- Current bladeburner action is shown on the character overview.
- Fix blackop being #000 on #000.

- The last clicked Tail Box goes in front of the others.
- Fixed an issue where some values were loaded as 0 when they should be null.
- Implemented toasts.
- .msg are no longer saved in the text file.
- Tail boxes no longer display all the args, they use “...” after 30 characters.
- Fixed cancelation penalty bonus not being properly applied after the IP <-> hostname switch.
- Fixed an exploit where you could send non-strings or numbers to other scripts.
- Fixed issue when trying to work for a faction with a work type that doesn't exist while already working for that faction.
- Fixed not being able to work for the CIA. (Don't ask)
- nerf noodle bar

1.8.13 v0.56.0 - 2021-10-11 Trimming the backlog (hydroflame & community)

** BREAKING **

- The ‘write’ function is now async. This helps when making scripts that write scripts.

** Terminal **

- ‘grow’ and ‘weaken’ have been added as terminal command. This should help player transition from commands to scripts. The tutorial also talks about it.
- ‘cp’ command added
- Improved performance by rate limiting refresh.

** IP vs Hostname **

- The game now uses hostname as primary key for it's servers (yeah believe it or not IPs were used until then). This has caused some issues with purchased servers (they couldn't be sold). You might need to soft reset for the game to fully convert itself.

** Sleeve **

- Fixed bug where they couldn't train at Volhaven.
- No longer consume all bonus time at once, making it look buggy.

** SF9 **

- Now boosts hacknet production by 8/12/14%

** Hacknet Servers **

- production nerfed by 10%
- Max money increase gets weaker above 10t max money

** Corporation **

- Warehouse tooltip now also displays the amount of space taken by products.
- Changed research box completely to avoid dependency on Treant (Treant is a pita)
- All textbox should accept MAX/MP case insensitive.
- Fixed export popup not refreshing dropdowns correctly.

- Fixed product mku becoming zero
- Increased scaling of Wilson to avoid feedback loop.
- Can no longer get in debt by buying real estate
- Bonus time is consumed faster.

**** Netscript ****

- isBusy takes bitverse and infiltration into account
- hospitalize can't be called when in infiltration.
- setToCommitCrime now accepts crime rough name instead of perfect name.
- disableLog All now works for bladeburner functions.
- Fixed netscript port for ns1.

**** Augmentation ****

- Added augmentation to Ti Di Hui that removes penalty for being unfocused.
- Neuroflux no longer appears in special factions.

**** Script Editor ****

- Ram check is debounced instead of refreshed every second.
- Added the vscode extension documentation to the game (it doesn't work well, thought)
- Fixed issue where autocomplete list would grow forever
- Added semi-monokai as theme.
- Fixed issue where modifying filename would mess it up.
- Font size can be changed now.

**** Infiltration ****

- Fixed issue where game controls would become unfocused.

**** Misc. ****

- Fixed loader incorrectly assuming some null values are incorrect.
- installBackdoor trigger Bitverse sequence
- Some improvements to the theme editor
- Improved documentation about where to learn javascript.
- Added some instructions for contributors.
- Fixed typo in corporation sell shares modal (@Saynt_Garmo)
- Fixed pagination being black on black in Active Scripts
- Create Script tab renamed to Script Editor
- Fixed an issue where corp some textbox wouldn't update when changing city.
- Fixed an issue where hacknet online time was always 0.
- Netscript function prompt fixed.
- Fixed miscalculation in growth.
- Script with syntax errors will try to be a tad more helpful.

- Corporations can no longer bribe bladeburners.
- Augmentation Graphene Branchblade renamed to Brachi, like the rest of them.
- All ram is displayed in GB/TB/PB now.
- Game now saves when saving a file, this can be turned off.
- Several improvement to log window.
- Bladeburner current action returns General type instead of the name of the action.
- Bladeburner travel and Sleeve travel respect disable ASCII.
- Tutorial fits on small screens.
- Import is much slower but more consistent now.
- Fix intelligence not updating properly.
- Added SF -1: Time Compression
- ReadTheDoc theme now matches the game.
- Logbox should wrap text better
- Logbox behavior should feel better.
- Fix font for AutoLink.exe
- nerf noodle bar

1.8.14 v0.55.0 - 2021-09-20 Material UI (hydroflame & community)

**** Global ****

- The game is now 100% in typescript, react, and Material-UI

**** Misc. ****

- Corporations can no longer bribe special factions
- Infiltration can no longer lose focus of the keyboard.
- Fix terminal line limit
- Added theme editor
- Theme applies on game load (@Nolshine)
- Sleeves no longer consume all bonus time for some actions
- Fix a bug where the autocomplete list would get duplicates
- Fix tutorial not scaling properly on small screens
- Import should be more consistent
- Typo with 'help' command
- Fix infinite loop in casino
- nerf noodle bar

1.8.15 v0.54.0 - 2021-09-20 One big react node (hydroflame & community)

** UI **

- The UI is now completely(ish) in react and I'm starting to implement Material-UI everywhere. This will help make the game feel more consistent.
- Major help from (@threehams)
- New Terminal
- New Active Scripts page
- New sidebar.
- New Character overview
- New tutorial
- New options page
- New create program page (@Nolshine)

** Netscript **

- Add companyName to getPlayer

** Factions **

- Megacorp factions are no longer removed when installing.

** Corporation **

- All research tooltips are always visible.
- Smart supply is enabled by default if purchased (@Nolshine)

** Misc. **

- Fix "Game saved" animation. (@Nolshine)
- Update commitCrime documentation (@Tryneus)
- Fix logbox scrolling weird (@Nolshine)
- Fix weird scrolling in corporations (@BartKoppelmans)
- Fix typo (@BartKoppelmans & @Nolshine)
- Delete game now has a confirmation modal (@Nolshine)
- Fix issue where skills would not get properly updated when entering new BN. (@Nolshine)
- Convert create gang to popup (@vmesecher)
- Fixed a bug that prevented travel to Sector-12 and New Tokyo when not using ASCII art.
- nerf noodle bar

1.8.16 v0.53.0 - 2021-09-09 Way too many things. (hydroflame & community)

** Dev? **

- The entire codebase has been run through a code prettifier, hurray for consistency. (@threehams)
- Lots of test. (@threehams)
- Massive improvements to build speed. (@threehams)

- Dev notes: This won't affect any players but is immensely useful for me.

** Hacknet **

- Converted to ts/react

** Resleeving **

- Converted to ts/react

** Sleeves **

- Converted to ts/react. The ui should also have a better feel.
- Fixed a bug that allowed players to recover shock much faster than intended.

** BN10 **

- You have access to Sleeves right away
- In BN10 Sleeves start with 75 shock and 25 sync.

** MathJax **

- Several tooltips have been updated to display the relevant formula in Mathjax, e.g. Favor and reputation

** Corporation **

- Completely rewritten in React. Paving the way for bigger change.
- Smart Supply is now smarter and won't deadlock the warehouse. It is also more configurable.
- Several UI fixes.

** Bladeburner **

- Action count is no longer decided when joining the Bladeburners. Experiences for all players should be more similar.

** Factions **

- No factions have home computer ram requirement. This caused some confusion for new players.

** Gang **

- Made it clear when there's a new equipment coming up.

** Netscript **

- `getActionCountRemaining` now returns Infinity for bladeburner general actions. (@brubsy)
- `getActionEstimatedSuccessChance` now returns 100% for Diplomacy and Hyperbolic Regeneration Chamber. (@brubsy)
- `disableLog('ALL')` now disables all logs individually, meaning you can re-enable the ones you want after. (@Cass)
- `getPlayer` returns `numPeopleKilled`.
- Dynamic RAM calculation errors have a better error message.
- Hide some functions from autocomplete.
- Added `getAugmentationPrice`, `getAugmentationRepReq`, deprecated `getAugmentationCost`. (@TempFound)
- Fixed bug where some crime API would return "assassinate" when that's not accepted in other functions.

** Coding Contract **

- Spiralize Matrix is easier to read.

**** Misc. ****

- The world map is now used in sleeve travel and bladeburner travel.
- noselect a bunch of stuff.
- Ascii maps letters are more contrasting
- Updated documentation for infiltration.
- Most money costs in the game will turn grey/cyan when you don't have enough money.
- Donation textbox has better look & feel.
- Tech vendors ram & cores buttons have better look and feels.
- cores cost modified to be a formula instead of a semi-random array of numbers.
- Tech vendors now give a hint about where to get bigger servers.
- logboxes now displays whitespaces exactly. (@Cass)
- nerf noodle bar

1.8.17 v0.52.9 - 2021-08-27 Less lag! (hydroflame & community)**** Active Scripts page ****

- Now less laggy, has pagination.

**** File diagnostic ****

- Added a popup found under options that shows the files you own and how large they are. This help find bugs and leftover massive logs files.

**** Corporation ****

- Added safeguard against a very specific bug that causes NaN money. I'm still not sure what the root cause is but it should prevent corp from breaking.

**** Netscript ****

- tprintf is a new function that doesn't print the filename.

**** Misc. ****

- Infiltration kills you if you try to automate it. (@threehams)
- Fix beautify button not working
- Added bladeburner_analysis_mult to getPlayer() (@brubsby)
- Fixed joining bladeburner via netscript functions. (@omuretsu)
- All bladeburner actions are click-to-copy
- nerf noodle bar

1.8.18 v0.52.8 - 2021-08-23 Fixing the previous patch tbh ROUND 2 (hydroflame)**** Script editor ****

- Correctly reloads old script when clicking "Script Editor"
- No longer jumps to the end of the text for no reason.

**** Hash upgrades ****

- Fixed an issue where the default option would say ecorp but was really foodnstuff

**** Misc. ****

- The “Delete all active script” button under the options has a clearer description.
- Removed some debug console.log
- nerf noodle bar

1.8.19 v0.52.7 - 2021-08-21 Fixing the previous patch tbh (hydroflame)

**** Netscript ****

- **API BREAKING CHANGE:** `getActionEstimatedSuccessChance` now returns a pair of value to reflect the UI changes. I’m very sorry.

**** Bladeburner ****

- General actions now display time required.
- Recruitment now displays success chance.
- All other success chance now display a range instead of a single value The real value is guaranteed to be within that range.

**** Misc. ****

- Fix tutorial not working after Monaco upate
- Fix logbox logs not taking up the whole logbox
- Fix script editor shortcut (ctrl+b)
- Fix Corporation popup appearing in the wrong order, hiding one of them
- Fix error when loading Corp
- Fix logbox dragging (smoother now)
- Fix logbox name collision
- Fix logbox allowing to open the same box multiple times
- Fix netscript write.
- nerf noodle bar

1.8.20 v0.52.6 - 2021-08-21 Logboxes and VS-code (hydroflame)

**** Text Editor ****

- Ace and Codemirror have been removed in favor of monaco (web version of vs-code). The options are a bit lackluster but more will be added as feedback comes.

**** Log boxes ****

- Multiple log boxes can be opened at once. They can be moved around the screen. (but the movement behavior is a bit weird.)

**** Misc. ****

- Job promotion now correctly updates the UI.

- Milestones now call the faction CyberSec instead of CSEC
- Can no longer create file that break the filesystem.
- Remove dollar sign in blade contract UI element
- nerf noodle bar

1.8.21 v0.52.5 - 2021-08-19 CPU cores are useful!? (hydroflame)

** Terminal **

- When executing 'run SCRIPT' any script can now add '-tail' to automatically bring up the logs.

** Netscript **

- The 'flags' function now works with single letter flags but they only take one dash.
- Fix several broken bladeburner netscript functions.
- Fix gang.getMemberInformation returning inconsistent data after the gang rework.

** CPU Cores **

- CPU Cores on the home computer now provide a bonus to grow() money gain and makes weaken lower more security. Only for scripts running on 'home'

** Misc. **

- Fix weird scrolling in the new Bladeburner React console.
- nerf noodle bar

1.8.22 v0.52.4 - 2021-08-19 Bladeburner in React (hydroflame)

** Bladeburner **

- The entire UI was rebuild in React. It should be more responsive

** Hacknet **

- Displays how many time each hash upgrade was bought.
- Displays cummulative effect of the upgrade.
- Removed "Close" button from hash upgrade menu.

** Misc. **

- More popup/modals have dark background, can be dismissed by clicking outside, or by pressing escape.
- Small reword in the guide.
- Fix several typos in the bladeburner documentation.
- Linting (no one cares except the dev)
- nerf noodle bar

1.8.23 v0.52.3 - 2021-08-15 Gangs were OP (hydroflame)

** Gang **

- Significant rework. Ascension is now based on exp gained.
- All upgrades give exp bonuses.
- Maximum gang members reduced to 12.
- Respect required to recruit sharply increased.
- Rewritten in React, the UI should be smoother and less laggy now.

** Infiltration **

- Now is Trusted protected.

** Misc. **

- Many UI element are now “noselect” protected.
- Fixed an issue where you could join the same faction twice via script and UI simultaneously.
- Factions list screen converted to React.
- nerf noodle bar

1.8.24 v0.52.2 - 2021-08-15 Oh yeah, BN11 is a thing (drunk hydroflame tbh)

** Source-Files **

- Source-File 11 now also provides a small reduction to the price increase multiplier.

** Augmentations **

- New Augmentation offered by Aevum, themed around 777 and offers some basic programs.
- Augmentation descriptions are now more concise and consistent.

** Misc. **

- nerf noodle bar

1.8.25 v0.52.1 - 2021-08-10 bugfixing (hydroflame & community)

** Misc. **

- Fix game crash/corruption when quitting a job while working for it unfocused.
- Fix typo in corporation Market Data.
- Fix typo in docs for hackPercent.
- The tutorial encourages the players to connect to home before creating *n00dles.script*
- The dark web *buy* command now accepts *-I* (one) and *-list* instead of just *-l*. Helps some confused players.
- Character overview screen no longer hidden on the corporation screen.
- Infiltration difficulty display is now more explicit (It’s a big arrow instead of just one word.)
- Fix wrong ram value in tutorial. (@MageKing17)
- Plenty of augmentation description cleanup (@Kwazygloo)

- Plenty of typo/description fixed (@MageKing17)
- Cleanup description of singularity function on readthedocs (@PurePandemonium)
- Fix bug when autolinking a server while backdooring (@schroederIT)
- nerf noodle bar

1.8.26 v0.52.0 - 2021-06-13 Infiltration 2.0 (hydroflame & community)

Infiltration

- Completely reworked. Not the same mechanic at all.

Terminal

- tail is smarter. It automatically assume the only possible options in some cases.

Intelligence

- Now available when starting BN5 instead of after beating it for the first time.
- Nerf the effect of intelligence on reputation gain.

Augmentation

- Added a new augmentation, the ‘Unstable Circadian Modulator’, whose gimmick is that its stats are randomized every hour.

Netscript

- ‘getPlayer’ is not a singularity function anymore.
- ‘hacknetNodes.constants’ returns the correct values.
- ‘createGang’ has been added.
- ‘inGang’ has been added.

Tutorial

- Updated the tutorial. Made it look cleaner, fixed typos, etc.

Misc.

- Fix many typos in literature (@kwazygloo)
- Fix being able to unfocus from gym and university.
- Fix being able to do hacking missions while unfocused.
- Fix many typos in Augmentation descriptions (@kwazygloo)
- More numbers handle absurdly large values. (@Tesseract1234567890)
- Fix many typos (@Tesseract1234567890)
- Fixed an issue that caused a UI desync when sleeves were set to workout stats other than strength at the gym.
- Fix weird alignment of donation text box and button. (@Tesseract1234567890)
- Fixed an issue where reputation could be transfered to new jobs when unfocused.
- Empty stack traces should no longer appear.
- Purchasing anything with Infinity money doesn’t result in NaN.
- nerf noodle bar

1.8.27 v0.51.10 - 2021-05-31 Focus Mark, Focus! (hydroflame)

Focus

- You can now use the terminal and write scripts while working for factions but you will gain reputation at a slower rate.

SF -1

- Added a new SF -1: Bypass

Gang

- “Vigilante justice”/”Ethical hacking” now reduces wanted level by a very small percentage as well an absolute value.

Netscript

- ‘tFormat’ now has a second argument to display with millisecond precision.
- ‘purchaseSleeveAug’ can no longer purchase the same aug over and over for the same sleeve.
- fix typo in logging for ‘getServerSecurityLevel’
- Fixed some weird issue where very rarely you would get 0 exp from ‘grow’
- ‘getActionTime’ now returns correct values for Diplomacy and Regeneration.

Corporations

- Fixed an exploit where you could get nearly infinite corporation funds by entering negative numbers in textboxes.
- Fixed an exploit where shares could be sold again by clicking the “sell share” button via scripts.

Documentation

- typo fix in purchaseTor
- typo fix in basicgameplay/stats

Misc.

- Very large number will no longer appear as “\$NaNt”
- Hash capacity now displays in the “big number” format.
- nerf noodle bar

1.8.28 v0.51.9 - 2021-05-17 offline progress and exports! (hydroflame & community)

Alias

- several commands can be included in 1 alias. Recursive alias now work to a depth of 10. (@Dawe)

Offline

- Offline money gain has been reworked (it is more generous)
- If you’re not working anywhere and go offline the game will work for you at all your factions evenly.

Export

- Exporting now gives +1 favor to all joined factions every 24h.

Corp

- Self-fund with an invalid name no longer takes away 150b anyway.

- Can no longer export negative amount

Bladeburner

- No longer waste overflowing time.

Text Editors

- All settings will now be saved and loaded correctly.

Terminal

- ‘scan’ now works for servers that are more than 21 character long.

Misc.

- ls now correctly lists all files.
- importing auto save+reloads (@Dawe)
- Fix a bug where .fconf could not be created
- Fix formatting inconsistencies for some logs of netscript functions.
- Fix a bug where Cashroot starter kit would appear as [object Object] in confirmation dialog.
- Fix some ram not displayed as 0.00GB
- Fix error message throw undefined variable error
- City hall now has some generic text if you can’t create a corp yet.
- Deleting a file without extension now returns an appropriate error message.
- Fixed an issue where bladeburner would miscalculate the cost of hospitalization.
- It is now possible to suppress bladeburner “action stopped” popup.
- Updated several dependencies (big who cares, I know)
- ls no longer prints lingering newline.
- Money earned/spent by sleeves is now tracked under Character>Money
- nerf noodle bar

1.8.29 v0.51.8 - 2021-05-07 It was there all along (hydroflame & community)

Servers

- Update n00dles metadata

Netscript

- ‘hashGainRate’ use the correct ‘usedRam’ and ‘maxRam’
- Fix ‘setActionAutolevel’ logging.
- Fix ‘setActionLevel’ not working at all.
- Add ‘installBackdoor’ singularity function.

Hacknet

- Fix Hacknet Servers total production always displaying 0

Documentation

- Updated guide to no longer recommend BN12.

- Fix documentation for maxNumNodes (@ModdedGamers)
- Fix typo in 'sourcefiles.rst'
- Fix typo in 'recommendedbitnodeorder.rst'
- Fix 'getServer' documentation missing 'server' argument.
- Fix missing ram cost in 'getData.rst'
- Fix basic formulas examples.
- Fix typo in BN11 description.
- Fix formatting issue in Bladeburner (@Pimgd)

Misc.

- Fix negative money being displayed in full.
- Fix Hacking Missions not working.
- Fix Corporation tree not rendering.
- Fix script being needlessly recompiled. This should save real ram (not game ram)
- w0rld_d43m0n can be backdoored
- Coding Contracts title is click-to-copy (@Rodeth)
- Covenant memory upgrade works better.
- Fix Neuroflux not being correctly calculated when entering BN with SF12.
- Delete Active Script now delete all active scripts, not just home.
- Now you can 'cd' in directories that only contain '.txt' files.
- Fix 'analyze' always saying players had root access
- Passive faction rep no longer builds for special factions.
- Donation option no longer appears for special factions.
- Rephrased some milestones.
- donation textbox now accepts money in the format '1b' and the like (@Dawe)
- Fix being able to join hated factions simultaneously. (@Dawe)
- 'ls' now displays files in multiple column. (Helps players with many files)
- Bladeburner multipliers now appear under Character>Stats and Character>Augmentation when they are relevant.
- Fix missing functions syntax highlight in codemirror.
- Fix infiltration number formatting.
- script income transfers to parent on death. This helps keep track of income for scripts that spawn short lived scripts.
- nerf noodle bar

1.8.30 v0.51.7 - 2021-04-28 n00dles (hydroflame & community)

Tutorial servers

- All the tutorial servers have been reverted to their original value

- The new server n00dles has been added as tutorial server.

Terminal

- ‘tail’ now accepts Pid.
- ‘analyze’ now handles Hacknet Servers correctly.
- ‘ServerProfiler.exe’ now handles Hacknet Servers correctly.

SF12

- Now makes you start with Neuroflux Governor equal to the level of the SF.

Netscript

- Deprecated ‘getServerRam’.
- ‘getServerMaxRam’ added to replace ‘getServerRam’
- ‘getServerUsedRam’ added to replace ‘getServerRam’
- ‘getBitnodeMultipliers’ is available inside BN5
- Time logged by hack/grow/weaken now displays in human time.
- thread count logged by hack/grow/weaken now displays with commas every thousands place.

Donation

- Always visible but locked until favor requirements are reached.

Augmentations

- City factions has been rebalanced to give a reason to visit them all.

Sleeves

- Fix sleeves not being able to work at Volhavens gym.

Lint

- This shouldn’t change anything but was like 10h of work. So I’m logging it.

Misc.

- Plethora of typo fixed (@Pimgd)
- ps documentation fix (@Dawe)
- The dev menu now has a quick bitflume option.
- Fix SF -1 not being as powerful as intended.
- Fix cashroot starter kit not displaying correctly.
- Fix DOM element ‘character-overview-text’ being nested twice.
- Hacknet documentation example fix.
- Money amount under 1000 dont display 3 decimal anymore.
- Fix nextSourceFile flag miscalculation on the bitverse (for Bn12)
- Faction invite text says “Decide later”/”Join!” instead of “No”/”Yes”
- nerf noodle bar

1.8.31 v0.51.6 - 2021-04-28 Backdoor! (hydroflame & community)

Backdoor

- a new terminal command, backdoor, has been added to help differentiate between the terminal hack command and the netscript hack function. (@dewint)

Servers

- foodnstuff, sigma-cosmetics, and joesguns have been rebalanced to help new players.

Milestones

- A new tab under the Help menu has been added to guide players through the game.

Casino

- Blackjack has been added (@BigD)

Netscript

- 'prompt' now converts input to JSON.
- 'getRunningScript' is a new netscript function that returns a bunch of data related to a running script.

Coding contracts

- trivial puzzles should no longer appear.

Infiltration

- All numbers are formatted like the rest of the game.

Misc.

- Server security is capped at 100.
- Added option to quit a job.
- 'cd' no longer works on unexistent folders.
- cd with no arguments brings you back to top level folder (@Andreas)
- 'softReset' documentation updated.
- Money tracker now accounts for going to the hospital manually.
- codemirror is now the default editor (for new save files)
- fix typo in dark web help text (@Rodeth)
- so many documentation and typos fixes (@Pimgd)
- A corruption visual effect has been added to location with servers that have backdoor installed. (@dewint)
- nerf noodle bar

1.8.32 v0.51.5 - 2021-04-20 Flags! (hydroflame)

Netscript

- 'flags' is a new function that helps script handle flags. This is subject to change if it doesn't meet the need of the players.
- 'ps' now returns the pid.
- 'tail' now works with pid as first argument.

- ‘tail’ hostname defaults to current server. (like the documentation says)
- ‘isRunning’ hostname defaults to current server.
- ‘isRunning’ now works with pid as first argument.

Gang

- Nerfed ascension mechanic once again :(

Misc.

- Souce-File typo fix
- Fix ‘while you were away’ screen.
- Bladeburner team size can no longer be set to negative amounts.
- nerf noodle bar

1.8.33 v0.51.4 - 2021-04-19 Manual hacking is fun (hydroflame)

Manual hacking

- These bonus require an install or a soft reset to take effect.
- Manual hacking gyms and university gives you a 10% discount.
- Manual hacking a corporation server decreases the penalty for leaving work early.

BladeBurner

- nerfed int exp gained.

Documentation

- purchaseServer specifies what happens on failure.
- Fixed typo in recommended bitnode page.
- Removed misleading ram requirements for hacking factions.

Netscript

- growthAnalyze handles Infinity correctly.

Misc.

- Faction Augmentation will list how much reputation is required even after that goal has been reached.
- Removed dollar sign in travel agency confirmation dialog box.
- Fixed typo in alpha-omega.lit
- the ‘Game saved!’ text no longer blocks the save game/options button.
- The text editor now remembers the location of your cursor and restores it.
- skills are recalculated instantly.
- Fix typo in Operation Zero description.
- nerf noodle bar

1.8.34 v0.51.3 - 2021-04-16 Y'all broke it on the first day (hydroflame)

Passive faction reputation

- Reworked, from 1 rep / 2 minute. Now is a complicated percentage of the reputation you'd gain working for them. It's not op but it feels a bit more useful.

Netscript

- print/tprint now take any number of arguments.
- print/tprint will now print object as json.
- print/tprint now handle passing in an undefined argument properly.

Casino

- Cannot bet negative money anymore.
- Roulette max bet is a bit higher.
- Coin Flip has a small cooldown.
- All buttons reject untrusted mouse events.

Documentation

- Changed a message that said nsjs only works on Chrome.

Bugfix

- hacknet.maxNumNodes now works for both nodes and servers.
- Fixed a bug where the popup boxes would contain data from previous popup boxes.
- .js files will also have the 'export async function' boilerplate.

Misc.

- turned off web form autocomplete for the terminal text input.
- Fixed an issue on Windows+Firefox where pressing up on the terminal would bring the cursor to the beginning of the line. (Issue #836)
- Hacknet node names is easier to handle for screen readers.
- Money spent on classes is now tracked independently of work money.
- running coding contract from the terminal will display its name.
- nerf noodle bar

1.8.35 v0.51.2 - 2021-04-09 Vegas, Baby! (hydroflame)

New location: The Iker Molina Casino

- A casino opened in Aevum. However the house is rumored to cheat. If only we could give them a taste of their own medicine.

Misc.

- Link to discord added under options
- 'getMemberInformation' doc updated, oops
- tech vendor now handle max ram and cores.
- nerf noodle bar

1.8.36 v0.51.1 - 2021-04-06 Bugfixes because the author of the last patch sucks (it's hydroflame)

Netscript

- 'getPlayer' returns players faction and tor
- 'hospitalization' is a new singularity function.
- 'gang.getMemberInformation' now returns more information.
- 'hacknet.hashCapacity' is a new hacknet function that returns the maximum hash capacity.

Hospitalization

- Now only cost at most 10% of your money.

Bugfix

- confirmation dialog box no longer use previous text

Accessibility

- The game is a little easier to handle for screen readers (yes, there's an absolute legend playing this game with a screen reader)
- Infiltration use buttons instead of a-links
- New option to disable ASCII art. This will make the metro map and world map display as a list of buttons.

Misc.

- 'fl1ght.exe' will no longer suggest the combat path. Related faction requirements unchanged.
- nerf noodle bar

1.8.37 v0.51.0 - 2021-03-31 Formulas (hydroflame)

Formulas API

- A new API is introduced, this gives players access to various formulas used in the game. It'll help you make more informed decisions.

Netscript

- 'getServer' is a new function meant to be used with the formulas API.
- 'getPlayer' is a new function meant to be used with the formulas API.
- 'getStats' and 'getCharacterInformation' are deprecated in favor of 'getPlayer'
- 'getCurrentServer' is a new function that returns the server the player is currently connected.

Display

- All money should now consistently be orange.
- All rep should now consistently be light-yellow.
- Most numbers should display consistently now (aka all money is formatted the same).

Click to copy

- Certain UI elements are now 'click-to-copy'

** Misc. **

- nerf noodle bar

1.8.38 v0.50.2 - 2021-03-25 Everyone asked for this one. (hydroflame)

BitNodeMultipliers

- ‘GangKarmaRequirements’: a new multiplier that influences how much karma is required to make a gang different bitnodes.

Netscript

- ‘connect’: a new singularity function that connects you to a server. (like the terminal command)
- ‘manualHack’: a new singularity function that performs a manual hack on the players current server.
- ns2 stack trace works on Firefox now.

Misc.

- New shortcut, Alt + b, brings you to bladeburner
- New shortcut, Alt + g, brings you to gang
- nerf noodle bar

1.8.39 v0.50.1 - 2021-03-22 (hydroflame)

Netscript

- getTaskStats works

Source-File -1

- Added a new Exploit

Factions

- Augmentations offered by a Faction but already bought are in a separate list at the bottom of the page.

Bug fixed

- Fixed a bug where completing a maxed non-repeatable BitNode would make its color on the BitVerse like level 1.

Misc.

- Minor spacing in stats tables.
- nerf noodle bar

1.8.40 v0.50.0 - 2021-03-20 Intelligence (hydroflame)

Intelligence

- int exp gain and effect has been reworked. It is now much more easy to acquire and far more powerful. The goal here is to feel like players have another tool in their arsenal.

Factions

- Hacking factions no longer have hacking level requirements since their associated servers do.

Misc.

- Sleeve styling.
- number formatting
- remove wiki button in Hacking Missions.
- Fix NaN displayed when very very large numbers are reached.
- nerf noodle bar

1.8.41 v0.49.2 - 2021-03-13 (hydroflame)

BN8

- A new bitnode multiplier has been added, it lets you reduce money from a server without gaining actually any money. This is important for BN8 where hack/grow can influence the stock market. No money can be gained from hacking but server money can still be reduced.

Documentation

- readthedocs should now be more consistent and many examples were added.

Netscript

- Ace editor will now correctly highlight all functions.
- 'tFormat' is a new netscript function that returns a human readable representation of milliseconds. eg. "2 hours 15 minute 43 seconds"

Gang

- style improvements

Bladeburner

- style improvements
- fix bug where 'skill list SKILL' would crash if skill is level 0.

Sleeve

- karma gain now scales with sync.

Misc.

- Fix issue where the effective stats under Character>Stats were being calculated.
- nerf noodle bar

1.8.42 v0.49.0 - 2021-03-11 Source-File -1 (hydroflame)

Source-File -1

- For advanced players: The game now embraces exploits and will reward players for doing so.

Gang

- ascension is less effective as the ascension multiplier goes up.
- territory gain scales with power difference.

Netscript

- 'gang.getEquipmentStats' returns the stats of the equipment.
- 'gang.getTaskStats' returns the stats of a task.

- ‘getCrimeStats’ returns the stats of a crime.
- Crashes should now print the ns stack trace.
- Log messages are now more consistent.
- ‘softReset’ now accepts a callback script like ‘installAugmentations’

Misc.

- Minor formatting under Hacking>Active Scripts
- option menu colors now match the rest of the game, kinda.
- nerf noodle bar

1.8.43 v0.48.0 - ASCII - 2021-03-07 (hydroflame)

ASCII

- Travel Agency now displays a world map
- Cities are now top view of metro station maps

Netscript

- **‘softReset’ is a new netscript function that performs a soft reset** regardless of if the player has bought augmentations or not.
- **‘getAugmentationStats’ is a new netscript function that returns the stats of** an augmentation.
- getCharacterInformation now additionally returns exp
- pid resets back to 1 when installing or destroying a BitNode.
- New ‘.ns’ scripts start with a main function.
- ‘hacknet.maxNumNodes’ returns the maximum number of hacknet nodes.

Bladeburner

- **Current stamina will scale as max stamina increases, this prevents players** from having very high penalty when they gain huge amount of exp at the start of a reset.

Misc.

- **Fixed an issue where SF3 was listed as infinitely repeatable and SF12 as** having a limit of 3.
- **Fixed an issue where the gang equipment screen would freeze the game if a** script installed augmentations while it is open.
- All BonusTime now displays in the ‘H M S’ format.
- Donation textbox style updated to match the rest of the game.
- Corporation name style updated to match the rest of the game.
- minor formatting under Hacking>Active Scripts
- typo in BN12 description
- BN12 now reduces contract money
- **Character>Stats percentages are aligned, server and hacknet limit are** displayed, if the player has SF5 the reduces stats are shown.
- **Character>Augmentations now displays by how much the player stats will** increase.

- **Character>Augmentations** has a badge indicating how many augs the player has bought but not installed
- **Character>Factions** has a badge indicating how many factions have pending invites.
- nerf noodle bar

1.8.44 v0.47.2 - 7/15/2019

Netscript Changes

- Added tail() Netscript function
- hacknet.getNodeStats() function now returns an additional property for Hacknet Servers: hashCapacity
- When writing to a file, the write() function now casts the data being written to a string (using String())
- BitNode-selection page now shows what Source-File level you have for each BitNode
- Overloaded kill() function so that you can kill a script by its PID
- spawn() now only takes 10 seconds to run (decreased from 20 seconds)
- **run() and exec() now return the PID of the newly-executed scripts, rather than a boolean**
 - (A PID is just a positive integer)
- run(), exec(), and spawn() no longer need to be await-ed in NetscriptJS
- Script parsing and RAM calculations now support ES9
- installAugmentations() no longer has a return value since it causes all scripts to die
- isBusy() now returns true if you are in a Hacking Mission
- Bug fix: workForFaction() function now properly accounts for disabled logs
- Bug fix: RAM should now be properly calculated when running a callback script with installAugmentations()
- Bug fix: Fixed bug that caused scripts killed by exit()/spawn() to “clean up” twice

Misc Changes

- The ‘kill’ Terminal command can now kill a script by its PID
- Added ‘Solarized Dark’ theme to CodeMirror editor
- After Infiltration, you will now return to the company page rather than the city page
- Bug fix: Stock Market UI should no longer crash for certain locale settings
- Bug fix: You can now properly remove unfinished programs (the *.exe-N%-INC files)
- Bug fix: Fixed an issue that allowed you to increase money on servers with a ‘maxMoney’ of 0 (like CSEC)
- Bug fix: Scripts no longer persist if they were started with syntax/import errors
- Bug fix: ‘hack’ and ‘analyze’ Terminal commands are now blocking
- Bug fix: Exp earned by duplicate sleeves at universities/gyms now takes hash upgrades into account

1.8.45 v0.47.1 - 6/27/2019

- **Stock Market changes:**
 - Transactions no longer influence stock prices (but they still influence forecast)
 - Changed the way stocks behave, particularly with regard to how the stock forecast occasionally “flips”

- Hacking & growing a server can potentially affect the way the corresponding stock's forecast changes
 - Working for a company positively affects the way the corresponding stock's forecast changes
- Scripts now start/stop instantly
- Improved performance when starting up many copies of a new NetscriptJS script (by Ornedan)
- Improved performance when killing scripts
- Dialog boxes can now be closed with the ESC key (by jaguilar)
- NetscriptJS scripts should now be “re-compiled” if their dependencies change (by jaguilar)
- write() function should now properly cause NetscriptJS scripts to “re-compile” (by jaguilar)

1.8.46 v0.47.0 - 5/17/2019

- **Stock Market changes:**
 - Implemented spread. Stock's now have bid and ask prices at which transactions occur
 - Large transactions will now influence a stock's price and forecast
 - This “influencing” can take effect in the middle of a transaction
 - See documentation for more details on these changes
 - Added getStockAskPrice(), getStockBidPrice() Netscript functions to the TIX API
 - Added getStockPurchaseCost(), getStockSaleGain() Netscript functions to the TIX API
- **Re-sleeves can no longer have the NeuroFlux Governor augmentation**
 - This is just a temporary patch until the mechanic gets re-worked
- hack(), grow(), and weaken() functions now take optional arguments for number of threads to use (by MasonD)
- codingcontract.attempt() now takes an optional argument that allows you to configure the function to return a contract's reward
- Adjusted RAM costs of Netscript Singularity functions (mostly increased)
- Adjusted RAM cost of codingcontract.getNumTriesRemaining() Netscript function
- Netscript Singularity functions no longer cost extra RAM outside of BitNode-4
- Corporation employees no longer have an “age” stat
- Gang Wanted level gain rate capped at 100 (per employee)
- Script startup/kill is now processed every 3 seconds, instead of 6 seconds
- getHackTime(), getGrowTime(), and getWeakenTime() now return Infinity if called on a Hacknet Server
- Money/Income tracker now displays money lost from hospitalizations
- Exported saves now have a unique filename based on current BitNode and timestamp
- Maximum number of Hacknet Servers decreased from 25 to 20
- Bug Fix: Corporation employees stats should no longer become negative
- Bug Fix: Fixed sleeve.getInformation() throwing error in certain scenarios
- Bug Fix: Coding contracts should no longer generate on the w0r1d_d43m0n server
- Bug Fix: Duplicate Sleeves now properly have access to all Augmentations if you have a gang

- Bug Fix: `getAugmentationsFromFaction()` & `purchaseAugmentation()` functions should now work properly if you have a gang
- Bug Fix: Fixed issue that caused messages (.msg) to be sent when refreshing/reloading the game
- Bug Fix: Purchasing hash upgrades for Bladeburner/Corporation when you don't actually have access to those mechanics no longer gives hashes
- Bug Fix: `run()`, `exec()`, and `spawn()` Netscript functions now throw if called with 0 threads
- Bug Fix: Faction UI should now automatically update reputation
- Bug Fix: Fixed `purchase4SMarketData()`
- Bug Fix: Netscript1.0 now works properly for multiple 'namespace' imports (`import * as namespace from "script"`)
- Bug Fix: Terminal 'wget' command now correctly evaluates directory paths
- Bug Fix: `wget()`, `write()`, and `scp()` Netscript functions now fail if an invalid filepath is passed in
- Bug Fix: Having Corporation warehouses at full capacity should no longer freeze game in certain conditions
- Bug Fix: Prevented an exploit that allows you to buy multiple copies of an Augmentation by holding the 'Enter' button
- Bug Fix: `gang.getOtherGangInformation()` now properly returns a deep copy
- Bug Fix: Fixed `getScriptIncome()` returning an undefined value
- Bug Fix: Fixed an issue with Hacknet Server hash rate not always updating

1.8.47 v0.46.3 - 4/20/2019

- Added a new Augmentation: The Shadow's Simulacrum
- Improved tab autocompletion feature in Terminal so that it works better with directories
- Bug Fix: Tech vendor location UI now properly refreshed when purchasing a TOR router
- Bug Fix: Fixed UI issue with faction donations
- Bug Fix: The money statistics & breakdown should now properly track money earned from Hacknet Server (hashes -> money)
- Bug Fix: Fixed issue with changing input in 'Minimum Path Sum in a Triangle' coding contract problem
- Fixed several typos in various places

1.8.48 v0.46.2 - 4/14/2019

- **Source-File 2 now allows you to form gangs in other BitNodes when your karma reaches a very large negative value**
 - (Karma is a hidden stat and is lowered by committing crimes)
- **Gang changes:**
 - Bug Fix: Gangs can no longer clash with themselves
 - Bug Fix: Winning against another gang should properly reduce their power
- Bug Fix: Terminal 'wget' command now works properly
- Bug Fix: Hacknet Server Hash upgrades now properly reset upon installing Augs/switching BitNodes

- Bug Fix: Fixed button for creating Corporations

1.8.49 v0.46.1 - 4/12/2019

- **Added a very rudimentary directory system to the Terminal**
 - Details here: <https://bitburner.readthedocs.io/en/latest/basicgameplay/terminal.html#filesystem-directories>
- Added numHashes(), hashCost(), and spendHashes() functions to the Netscript Hacknet Node API
- ‘Generate Coding Contract’ hash upgrade is now more expensive
- ‘Generate Coding Contract’ hash upgrade now generates the contract randomly on the server, rather than on home computer
- The cost of selling hashes for money no longer increases each time
- Selling hashes for money now costs 4 hashes (in exchange for \$1m)
- Bug Fix: Hacknet Node earnings should work properly when game is inactive/offline
- Bug Fix: Duplicate Sleeve augmentations are now properly reset when switching to a new BitNode

1.8.50 v0.46.0 - 4/3/2019

- Added BitNode-9: Hacktocracy
- Changed BitNode-11’s multipliers to make it slightly harder overall
- Source-File 11 is now slightly stronger
- Added several functions to Netscript Sleeve API for buying Sleeve augmentations (by hydroflame)
- Added a new stat for Duplicate Sleeves: Memory
- Increase baseline experience earned from Infiltration, but it now gives diminishing returns (on exp) as you get to higher difficulties/levels
- In Bladeburner, stamina gained from Hyperbolic Regeneration Chamber is now a percentage of your max stamina
- **Corporation Changes:**
 - ‘Demand’ value of products decreases more slowly
 - Bug Fix: Fixed a Corporation issue that broke the Market-TA2 Research
 - Bug Fix: Issuing New Shares now works properly
- Bug Fix: Money Statistics tracker was incorrectly recording profits when selling stocks manually
- Bug Fix: Fixed an issue with the job requirement tooltip for security jobs

1.8.51 v0.45.1 - 3/23/2019

- Added two new Corporation Researches
- General UI improvements (by hydroflame and koriar)
- Bug Fix: Sleeve Netscript API should no longer cause Dynamic RAM errors
- Bug Fix: sleeve.getSleeveStats() should now work properly

1.8.52 v0.45.0 - 3/22/2019

- **Corporation changes:**
 - Decreased the time of a full market cycle from 15 seconds to 10 seconds.
 - This means that each Corporation ‘state’ will now only take 2 seconds, rather than 3
 - Increased initial salaries for newly-hired employees
 - Increased the cost multiplier for upgrading office size (the cost will increase faster)
 - The stats of your employees now has a slightly larger effect on production & sales
 - Added several new Research upgrades
 - Market-TA research now allows you to automatically set sale price at optimal values
 - Market-TA research now works for Products (not just Materials)
 - Reduced the amount of Scientific Research needed to unlock the Hi-Tech R&D Laboratory from 10k to 5k
 - Energy Material requirement of the Software industry reduced from 1 to 0.5
 - It is now slightly easier to increase the Software industry’s production multiplier
 - Industries now have a maximum number of allowed products, starting at 3. This can be increased through research.
 - You can now see an approximation of how each material affects an industry’s production multiplier by clicking the “?” help tip next to it
 - Significantly changed the effects of the different employee positions. See updated descriptions
 - Reduced the amount of money you gain from private investors
 - Training employees is now 3x more effective
 - Bug Fix: An industry’s products are now properly separated between different cities
- The QLink Augemntation is now significantly stronger, but also significantly more expensive (by hydroflame)
- Added a Netscript API for Duplicate Sleeves (by hydroflame)
- Modified the multipliers of BitNode-3 and BitNode-8 to make them slightly harder
- After installing Augmentations, Duplicate Sleeves will now default to Synchronize if their Shock is 0
- Bug Fix: Bladeburner’s Hyperbolic Regeneration Chamber should no longer instantly refill all stamina
- Bug Fix: growthAnalyze() function now properly accounts for BitNode multipliers
- Bug Fix: The cost of purchasing Augmentations for Duplicate Sleeves no longer scales with how many Augs you’ve purchased for yourself

1.8.53 v0.44.1 - 3/4/2019

- **Duplicate Sleeve changes:**
 - You can now purchase Augmentations for your Duplicate Sleeves
 - Sleeves are now assigned to Shock Recovery task by default
 - Shock Recovery and Synchronize tasks are now twice as effective

- Changed documentation so that Netscript functions are own their own pages. Sorry if this is annoying, it was necessary for properly cross-referencing
- Officially deprecated the Wiki (the fandom site). Use the 'readthedocs' Documentation instead
- Bug Fix: 'rm' Terminal and Netscript commands now work on non-program files that have '.exe' in the name (by Github user MasonD)
- Bug Fix: The 'Find All Valid Math Expressions' Coding Contract should now properly ignore whitespace in answers
- Bug Fix: The 'Merge Overlapping Intervals' Coding Contract should now properly accept 2D arrays when being attempted through Netscript

1.8.54 v0.44.0 - 2/26/2019

- **Bladeburner Changes:**
 - Reduced the amount of rank needed to earn a skill point
 - Reduced the effects of the “Reaper” and “Evasive System” skills
 - Increased the effect of the “Hyperdrive” and “Hands of Midas” skills
 - Slightly increased the rate which the skill point cost rises for almost all skills
 - The “Overlock” Skill now has a maximum level of 90 instead of 95
 - Money earned from Contracts increased by 400%
 - Changed the way population affects success rate. Extreme populations now have less dramatic effects
 - Added two new General Actions: Diplomacy and Hyperbolic Regeneration Chamber
 - Lowered the rep and money cost of the “Blade’s Simulacrum” augmentation
 - Significantly decreased the initial amount of Contracts/Operations (the “Contracts/Operations remaining” value)
 - Decreased the rate at which the amount of Contracts/Operations increases over time
 - Decreased the number of successes you need to increase the max level of a Contract/Operation
 - Increased the average number of Synthoid communities each city has
 - Reduced the amount by which a successful raid will decrease the population of a city
 - The “riots” event will now increase the chaos of a city by a greater amount
 - Significantly increased the effect that Agility and Dexterity have on action time
- **Added new BitNode multipliers:**
 - HomeComputerRamCost - Affects how much it costs to upgrade home computer’s RAM
 - DaedalusAugsRequirement - Affects how many Augmentations you need in order to get invited to Daedalus
 - FourSigmaMarketDataCost - Affects how much it costs to unlock the stock market’s 4S Market Data
 - FourSigmaMarketDataApiCost - Affects how much it costs to unlock the stock market’s 4S Market Data API
- A few minor changes to BitNode multipliers across the board (mostly for the new multipliers)
- ‘The Covenant’ now requires 20 total Augmentations to get invited, rather than 30

- You can now purchase permanent Duplicate Sleeves from ‘The Covenant’. This requires Source-File 10, and you must be in BN-10 or after
- You can now track where all of your money comes from in the ‘Stats’ page
- Increased the money gained from Coding Contracts by 50%
- `getCharacterInformation()` function now returns the player’s HP and max HP
- Bug Fix: You can no longer disconnect the enemy’s connections in Hacking Missions
- Bug Fix: Duplicate Sleeve faction reputation gain is now properly affected by faction favor
- Bug Fix: After installing Augmentations, the Terminal display will now correctly show the current server as “home”
- Bug Fix: Fixed an exploit where you could change the duration of timed functions (e.g. `hack`, `weaken`) in NetscriptJS
- Bug Fix: You should now properly be able to use the `ServerProfile.exe` program
- Bug Fix: Prevented exploit that allowed you to accept faction invites programmatically through NetscriptJS
- Bug Fix: Faction invitations for megacorporations should now work properly

1.8.55 v0.43.1 - 2/11/2019

- **Terminal changes:**
 - Quoted arguments are now properly parsed. (e.g. `run f.script “this is one argument”` will be correctly parsed)
 - Errors are now shown in red text
 - `unalias` command now has a different format and no longer needs the quotations
 - Bug Fix: Fixed several edge cases where autocomplete wasn’t working properly
- Added two new Bladeburner skills for increasing money and experience gain
- Made some minor adjustments to Bladeburner UI
- Corporation “Smart Factories” and “Smart Storage” upgrades have slightly lower price multipliers
- Added `nFormat` Netscript function
- Added 6 new Coding Contract problems
- Updated documentation with list of all Coding Contract problems
- Minor improvements for ‘Active Scripts’ UI
- Implemented several optimizations for active scripts. The game should now use less memory and the savefile should be slightly smaller when there are many scripts running
- Bug Fix: A Stock Forecast should no longer go above 1 (i.e. 100%)
- Bug Fix: The cost of Resleeves should no longer be affected by buying Augs
- Bug Fix: Duplicate Sleeves now use their own stats to determine crime success rate, instead of the host consciousness’ stats
- Bug Fix: You can now call the `prompt()` Netscript function from multiple scripts simultaneously

1.8.56 v0.43.0 - 2/4/2019

- Added BitNode-10: Digital Carbon
- **Stock Market Changes:**
 - Each stock now has a maximum number of shares you can purchase (both Long and Short positions combined)
 - Added `getStockMaxShares()` Netscript function to the TIX API
 - The cost of 4S Market Data TIX API Access increased from \$20b to \$25b
- **Job Changes:**
 - You can now hold multiple jobs at once. This means you no longer lose reputation when leaving a company
 - Because of this change, the `getCharacterInformation()` Netscript function returns a slightly different value
- **Script Editor Changes:**
 - Added new script editor: CodeMirror. You can choose between the old editor (Ace) or CodeMirror
 - Navigation keyboard shortcuts no longer work if the script editor is focused
- Trying to programmatically run a script (`run()`, `exec()`) with a ‘threads’ argument of 0 will now cause the function to return false without running the script
- Home Computer RAM is now capped at 2^{30} GB (1073741824 GB)
- The maximum amount, maximum RAM, and cost of purchasing servers can now vary between different BitNodes (new BitNode multipliers)
- Pop-up dialog boxes are a little bit bigger
- Bug Fix: When importing scripts, “./” will now be properly ignored (e.g. `import { foo } from “./lib.script”`)

1.8.57 v0.42.0 - 1/8/2019

- **Corporation Changes:**
 - Corporation can now be self-funded with \$150b or using seed money in exchange for 500m newly-issued shares
 - In BitNode-3, you no longer start with \$150b
 - Changed initial market prices for many materials
 - Changed the way a material’s demand, competition, and market price change over time
 - The sale price of materials can no longer be marked-up as high
 - Added a Research Tree mechanic. Spend Scientific Research on permanent upgrades for each industry
 - You can now redistribute earnings to shareholders (including yourself) as dividends
 - Cost of “Smart Supply” upgraded reduced from \$50b to \$25b
 - Now has offline progress, which works similarly to the Gang/Bladeburner mechanics
 - Slightly reduced the amount of money offered to you by investment firms
 - Employee salaries now slowly increase over time

- Slightly reduced the effect “Real Estate” has on the Production Multiplier for the Agriculture industry
- Changed the way your Corporation’s value is calculated (this is what determines stock price)
- After taking your corporation public, it is now possible to issue new shares to raise capital
- Issuing new shares can only be done once every 12 hours
- Buying back shares must now be done at a premium
- Selling shares can now only be done once per hour
- Selling large amounts of shares now immediately impacts stock price (during the transaction)
- Reduced the initial cost of the DreamSense upgrade from \$8b to \$4b, but increased its price multiplier
- Reduced the price multiplier for ABC SalesBots upgrade
- Added getOrders() Netscript function to the TIX API
- Added getAugmentationPrereq() Singularity function (by havocmayhem)
- Added hackAnalyzePercent() and hackAnalyzeThreads() Netscript functions
- Stock Market, Travel, and Corporation main menu links are now properly styled
- Many pop-up/dialog boxes now support the ‘Enter’ and ‘Esc’ hotkeys. If you find a pop-up/dialog box that doesn’t support this, let me know specifically which one (‘Enter’ for the default option, ‘Esc’ for cancelling and closing the pop-up box)
- Added “brace_style = preserve_inline” configuration to Script Editor Beautifier
- ServerProfiler.exe can now be purchased from the Dark Web
- Added an option to copy save data to clipboard
- Added total multiplier information on the “Augmentations” page
- Bug Fix: gymWorkout() Singularity function should now work properly with Millenium Fitness Gym
- Began migrating gameplay information to the ReadTheDocs documentation

1.8.58 v0.41.2 - 11/23/2018

- **IMPORTANT - Netscript Changes:**
 - rm() now takes an optional parameter that lets you specify on which server to delete the file
 - Added growthAnalyze() Netscript function
- **Gang Changes:**
 - UI now displays your chance to win a clash with other gangs
 - Added getChanceToWinClash() function to the Gang API
 - Added getEquipmentType() function to the Gang API
 - Added several new hacking-based equipment and Augmentations
 - Rebalanced several equipment/upgrades to give less defense
 - Wanted level gain rate is now be slightly higher for all tasks
 - Rebalanced parameters for “hacking” tasks
- Added new Main Menu configuration in .fconf: “compact”
- Added the terminal command ‘expr’, which can be used to evaluate simple mathematical expressions

- Bug Fix: Can no longer purchase duplicate equipment/Augmentations through `gang.purchaseEquipment()`
- Bug Fix: `scp()` should no longer throw errors when used with 2-arguments and an array of files
- Bug Fix: Coding Contracts no longer give money in BitNode-8
- Bug Fix: In Bladeburner, you can no longer start a BlackOp through the Netscript API if it has already been completed
- Bug Fix: In Bladeburner, fixed a bug which caused the configured 'automate' actions to occasionally be switched to other actions
- Bug Fix: 'Return to World' button at locations no longer accumulates event listeners
- Bug Fix: Working & taking classes now continuously add/subtract money during the action, instead of doing it at completion
- Bug Fix: Top-right overview panel now displays negative money using '-' instead of '()
- Bug Fix: Stock Market UI should no longer show 'NaN' profit immediately after buying a stock

1.8.59 v0.41.1 - 11/5/2018

- **IMPORTANT - Netscript Changes:**

- `purchaseTor()` now returns true if you already have a TOR router (it used to return false)
- `getPurchasedServerCost()` now returns Infinity if the specified RAM is an invalid amount or is greater than the max amount of RAM (2^{20} GB)
- Added `purchase4SMarketData()` and `purchase4SMarketDataTixApi()` functions
- `getScriptLogs()` now takes in optional arguments that let you get the logs of another script

- **Stock Market changes:**

- Stocks now have "maximum prices". These are hidden from the player
- If a stock reaches its "maximum price", it will most likely drop in value (although it might still rise)
- Each stock has its own, unique maximum price
- Maximum price for each stock are randomly generated and change during each 'reset'
- Stock Market cycles are now accumulated/stored, much like it is for Gangs and Bladeburners
- **Accumulated/stored cycles cause stock prices to update up to 50% faster (from every 6 seconds to 4 seconds)**

* This means that after coming back from being offline, stock prices will update faster to make up for offline time

- Decreased the Hacking Level multiplier for BitNodes 6 and 7 to 0.4 (from 0.5)
- Bladeburner console history is now saved and persists when switching screens or closing/reopening the game
- In Bladeburner, if your stamina reaches 0 your current action will be cancelled
- `b1t_flum3.exe` is no longer removed from your home computer upon reset
- Added main menu link for the Stock Market (once you've purchased an account)
- Job main menu link only appears if you actually have a job
- Bug Fix: Netscript Gang API functions `purchaseEquipment()` and `ascendMember()` should now work properly
- Bug Fix: After installing Augs, the "Portfolio Mode" button on the Stock Market page should be properly reset

- Bug Fix: bladeburner.getActionCountRemaining()'s return value is now rounded down (by Kline-)

1.8.60 v0.41.0 - 10/29/2018

- WARNING: In NetscriptJS, defining a function called print() is no longer possible
- **Gang Mechanic Changes (BitNode-2):**
 - Added a Gang Netscript API
 - Added new 'ascension' mechanic for Gang Members
 - The first three gang members are now 'free' (can be recruited instantly)
 - Maximum number of increased Gang Members increased from 20 to 30
 - Changed the formula for calculating respect needed to recruit the next gang member
 - Added a new category of upgrades for Gang Members: Augmentations
 - Non-Augmentation Gang member upgrades are now significantly weaker
 - Reputation for your Gang faction can no longer be gained through Infiltration
 - Re-worked the territory 'warfare' mechanic so that player can choose when to engage in it
 - Gang Members can now be killed during territory 'warfare'
 - Changed BitNode-2 Multipliers to make hacking slightly less profitable
 - Gang Member Equipment + Upgrades now get cheaper as your gang grows in power and respect
 - The effects of Source-File 2 are now slightly more powerful
- RAM Cost of accessing the global document object lowered from 100 GB to 25 GB
- RAM Cost to use Singularity Functions outside of BitNode-4 lowered by 75%. They now only cost twice as much as they do in BitNode-4
- b1t_flum3.exe now takes significantly less time to create
- Crimes committed through Singularity function no longer give half money/exp (there is now no penalty)
- Improved number formatting for Player 'work' actions (including crimes, etc.). These numbers should also adhere to locale settings now (by Kline-)
- The order that Augmentations are listed in (when purchasing from Faction and viewing your Augmentations) is now saved and persists when choosing different orders
- getCharacterInformation() Singularity function now returns multiplier information (from Augmentations/Source Files)
- Bug Fix: Calling print() in NetscriptJS no longer brings up the print dialog
- Bug Fix: Fixed a bug that sometimes caused a blank black screen when destroying/resetting/switching BitNodes
- Bug Fix: Netscript calls that throw errors will now no longer cause the 'concurrent calls' error if they are caught in the script. i.e. try/catch should now work properly in scripts
- Bug Fix: Fixed a bug where sometimes the NeuroFlux Governor Augmentation level would be incorrectly calculated when the game was loaded
- Bug Fix: Fixed a bug where calling the scp() Netscript function with invalid hostname/ips would throw an unclear error message
- Bug Fix: Bladeburner API function getActionCountRemaining() should now work properly for BlackOps

- Bug Fix: Black Ops can no longer be attempted out-of-order or without the required rank via Bladeburner API
- Bug Fix: Dynamic RAM Calculation now properly accounts for number of threads
- RAM cost for basic Netscript functions added to documentation (by CBJamo)

1.8.61 v0.40.5 - 10/09/2018

- Added `codingcontract.getContractType()` Netscript function
- Bug Fix: `codingcontract.getData()` Netscript function now returns arrays by value rather than reference
- Bug Fix: Decreased highest possible data value for 'Find Largest Prime Factor' Coding Contract (to avoid hangs when solving it)
- Bug Fix: Fixed a bug that caused game to freeze during Coding Contract generation

1.8.62 v0.40.4 - 9/29/2018

- Added new Coding Contracts mechanic. Solve programming problems to earn rewards
- The `write()` and `read()` Netscript functions now work on scripts
- Added `getStockSymbols()` Netscript function to the TIX API (by InfraK)
- Added `wget()` Netscript function
- Added `bladeburner.getActionRepGain()` function to the Netscript Bladeburner API
- The `getLevelUpgradeCost()`, `getRamUpgradeCost()`, and `getCoreUpgradeCost()` functions in the Hacknet API now return Infinity if the node is at max level. See documentation
- It is now possible to use freely use angled bracket (<, >) and create DOM elements using `tprint()`
- The game's theme colors can now be set through the Terminal configuration (`.fconf`).
- You can now switch to the old left-hand main menu bar through the Terminal configuration (`.fconf`)
- Bug Fix: `grow()` percentage is no longer reported as Infinity when a server's money is grown from 0 to X
- Bug Fix: Infiltration popup now displays the correct amount of exp gained

1.8.63 v0.40.3 - 9/15/2018

- **Bladeburner Changes:**
 - Increased the effect that agi and dexterity have on action time
 - Starting number of contracts/operations available will be slightly lower
 - Random events will now happen slightly more often
 - Slightly increased the rate at which the Overclock skill point cost increases
- The maximum volatility of stocks is now randomized (randomly generated within a certain range every time the game resets)
- Increased the range of possible values for initial stock prices
- `b1t_flum3.exe` program can now be created immediately at Hacking level 1 (rather than hacking level 5)
- UI improvements for the character overview panel and the left-hand menu (by mat-jaworski)

- General UI improvements for displays and Terminal (by mat-jaworski)
- Added optional parameters to the getHackTime(), getGrowTime(), and getWeakenTime() Netscript functions
- Added isLogEnabled() and getScriptLogs() Netscript functions
- Added donateToFaction() Singularity function
- Updated documentation to reflect the fact that Netscript port handles (getPortHandle()) only works in NetscriptJS (2.0), NOT Netscript 1.0
- Added tryWrite() Netscript function
- When working (for a company/faction), experience is gained immediately/continuously rather than all at once when the work is finished
- Added a setting in .conf for enabling line-wrap in the Terminal input
- Adding a game option for changing the locale that most numbers are displayed in (this mostly applies for whenever money is displayed)
- The randomized parameters of many high-level servers can now take on a higher range of values
- Many ‘foreign’ servers (hackable servers that you don’t own) now have a randomized amount of RAM
- Added ‘wget’ Terminal command
- Improved the introductory tutorial

1.8.64 v0.40.2 - 8/27/2018

- **Bladeburner Changes:**
 - Added getBonusTime(), getSkillUpgradeCost(), and getCity() Netscript functions to the API
 - Buffed the effects of many Bladeburner Augmentations
 - The Blade’s Simulacrum Augmentation requires significantly less reputation but slightly more money
 - Slightly increased the amount of successes needed for a Contract/Operation in order to increase its max level
 - Increased the amount of money gained from Contracts by ~25%
 - Increased the base amount of rank gained from Operations by 10%
 - Significantly increased the ‘randomness’ in determining a Contract/Operation’s initial count and rate of count increase
 - The number (count) of Operations should now increase significantly faster
 - There are now, on average, more Synthoid communities in a city
 - If automation is enabled (the feature in Bladeburner console), then switching to another action such as working for a company will now disable the automation
- **Stock Market Changes:**
 - Added a watchlist filter feature to the UI that allows you to specify which stocks to show
 - Added the Four Sigma (4S) Market Data feed, which provides volatility and price forecast information about stocks
 - Added the 4S Market Data TIX API, which lets you access the aforementioned data through Netscript
- There is now a setting for enabling/disabling the popup that appears when you are hospitalized

- Bug Fix: Stock market should now be correctly initialized in BitNode-8 (by Kline-)
- Bug Fix: bladeburner.getCurrentAction() should now properly an 'Idle' object rather than null (by Kline-)
- Bug Fix: Bladeburner skill cost multiplier should now properly increase in BitNode-12 (by hydroflame)
- Bug Fix: 'document', 'hacknet', and 'window' keywords should no longer be counted multiple times in RAM calculations
- Bug Fix: Joining factions through Singularity functions should now prevent you from joining opposing factions
- Bug Fix: Four Sigma should no longer have two 'Speech Enhancement' Augmentations (by Kline-)

1.8.65 v0.40.1 - 8/5/2018 - Community Update

- Added getPurchasedServerCost() Netscript function (by kopelli)
- Added getFavorToDonate() Netscript function (by hydroflame)
- Added getFactionFavorGain() and getCompanyFavorGain() Singularity functions (by hydroflame)
- Accumulated 'bonus' time in Bladeburner is now displayed in the UI (by hydroflame)
- The Red Pill can now be purchased with negative money (since its supposed to be free) (by hydroflame)
- Cranial Signal Processor Augmentations now have the previous generation as a prerequisite. i.e. Cranial Signal Processor - Gen II requires Gen I (by Kline-)
- Terminal now supports semicolon usage (end of command). This allows chaining multiple Terminal commands (by hydroflame)
- Bladeburner Raid operations can no longer be performed if your estimate of Synthoid communities is zero (by hydroflame)
- The difficulty of BN-12 now scales faster (by hydroflame)
- Active Scripts UI now shows a RAM Usage bar for each server (by kopelli)
- Bug Fix: Corrected terminal timestamp format (by kopelli)
- Bug Fix: NetscriptJS scripts should now die properly if they don't have a 'main' function (by hydroflame)
- Bug Fix: write(), read(), and tryWrite() Netscript functions should now work properly for writing Arrays/objects to Netscript Ports
- Various minor UI/QOL fixes by hydroflame, kopelli, and Kline-

1.8.66 v0.40.0 - 7/28/2018

- **WARNING: This update makes some significant changes to Netscript and therefore you may need to make some changes to your scripts. See [this post](#) [this post](#) for details**
- Netscript 1.0 (NS1) now uses a fully-fledged ES5 JavaScript Interpreter. This means many new features are now available in NS1, and this also fixes several bugs. However this also means any ES6+ features are no longer supported in NS1
- When a server is hacked with a very large number of threads and left with no money, the server's security level now only increases by however many threads were needed to drain the server. For example, if you hack a server with 5000 threads but it only needed 2000 threads to deplete the server's money, then the server's security will only increase as if you had hacked it with 2000 threads (change by hydroflame)
- Added getCurrentAction() to Bladeburner API

- Added a variety of functions to Bladeburner API that deal with action levels (change by hydroflame)
- Added `getPurchasedServerLimit()` and `getPurchasedServerMaxRam()` functions to Netscript (change by hydroflame & kopelli)
- Added `getOwnedSourceFiles()` Singularity function (by hydroflame)
- Completely re-designed the Hacknet Node API
- `getSkillLevel()` in Bladeburner API now returns an error if no argument is passed in (as opposed to an object with all skill levels). This may break scripts
- Minimum Netscript execution time reduced from 15ms to 10ms (configurable in Options)
- Company reputation needed to get invited to Megacorporation factions decreased from 250k to 200k
- HP is now reset (restored) when Augmenting
- Source-File 6 now increases both the level and experience gain of all combat stats (it was only experience gain previously)
- Reverted a previous change for Source-File 12. It's benefits are now multiplicative rather than additive
- Starting Infiltration security level for almost every location decreased by ~10%
- Changed 'flight.exe' message when its listed conditions are fulfilled (by hydroflame)
- The 'Save Game' button in the top-right overview panel now flashes red if autosave is disabled
- Bug Fix: Infiltration buttons can no longer be clicked through NetscriptJS
- Bug Fix: Bladeburner 'Overclock' skill can no longer be leveled above max level through the API (by hydroflame)
- Bug Fix: Healthcare division in Bladeburner should no longer cause game to crash

1.8.67 v0.39.1 - 7/4/2018

- Bladeburner Rank gain in BN-7 is now reduced by 40% instead of 50%
- Quadrupled the amount of money gained from Bladeburner contracts
- Added `joinBladeburnerDivision()` Netscript function to Bladeburner API
- Doubled the effects of Source-File 5. Now gives 8%, 12%, and 14% increase to all hacking multipliers at levels 1, 2, and 3, respectively (increased from 4%/6%, 7%)
- Increased the effect of Source-File 8. It now gives a 12%, 18% and 21% to your hacking growth multiplier at levels 1, 2, and 3, respectively (increased from 8%, 12%, 14%)
- The effect of Source-File 12 is now additive with itself, rather than multiplicative. This means that level N of Source-File 12 now increases all multipliers by N%
- The setting to suppress the confirmation box when purchasing Augmentations was moved into the main Options menu (by Github user hydroflame)
- Bug Fix: Crime Success rates were being calculated incorrectly (by Github user hydroflame)
- When an Infiltration is finished, you will now return back to the company's page, rather than the city
- Infiltration faction reputation selector now remembers your last choice
- Significantly increased the amount of money gained from Infiltration
- Bug Fix: Copying a NetscriptJS script to another server using `scp` now properly takes into account the script's changes.

- Bug Fix: Fixed an issue where game would not load in Edge due to incompatible features
- `travelToCity()` Singularity function no longer grants Intelligence exp”

1.8.68 v0.39.0 - 6/25/2018

- Added BitNode-7: Bladeburner 2079
- Infiltration base difficulty decreased by 10% for most locations
- Experience gains from Infiltration slightly increased
- Money gained from Infiltration increased by 20%
- Added ‘var’ declarations in Netscript 1.0 (only works with ‘var’, not ‘let’ or ‘const’)
- Script base RAM cost is now 1.6 GB (increased from 1.4 GB)
- While/for loops and if statements no longer cost RAM in scripts
- Made short-circuit evaluation logic more consistent in Netscript 1.0 (see <https://github.com/danielyxie/bitburner/issues/308>)
- Changelog button in the Options menu now links to the new Changelog URL (by Github user thePalindrome)
- Skill level calculation is now ‘smoother’ (by Github user hydroflame)
- Added a button to ‘beautify’ scripts in the text editor (by Github user hydroflame)
- Added favicon (by Github user kopelli)

1.8.69 v0.38.1 - 6/15/2018

- Bug Fix: Using ‘Object.prototype’ functions like `toLocaleString()` or `toString()` should no longer cause errors in NetscriptJS
- **Implemented by Github user hydroflame:**
 - Accessing the ‘window’ and ‘document’ objects in Netscript JS now requires a large amount of RAM (100 GB)
 - Added game option to suppress travel confirmation
 - Text on buttons can no longer be highlighted
 - Bug Fix: Fixed an issue that caused NaN values when exporting Real Estate in Corporations
 - Bug Fix: Competition and Demand displays in Corporation are now correct (were reversed before)
 - Added `ps()` Netscript function
 - Bug Fix: `grow()` should no longer return/log a negative value when it runs on a server that’s already at max money
 - Bug Fix: `serverExists()` Netscript function should now properly return false for non-existent host-name/ips
 - Bug Fix: Sever’s security level should now properly increase when its money is grown to max value

1.8.70 v0.38.0 - 6/12/2018

- New BitNode: BN-12 The Recursion - Implemented by Github user hydroflame
- **Bladeburner Changes:**
 - Bladeburner progress is no longer reset when installing Augmentations
 - The number of successes needed to increase a Contract/Operation's max level now scales with the current max level (gradually gets harder)
 - All Bladeburner Augmentations are now slightly more expensive and require more reputation
 - Black Operations now give higher rank rewards
 - Doubled the base amount of money gained from Contracts
 - Increased the amount of experience gained from Contracts/Actions
 - Added a new Augmentation: The Blade's Simulacrum
 - Bladeburner faction reputation gain is now properly affected by favor
- Hacking is now slightly less profitable in BitNode-3
- Updated Hacknet Nodes UI - Implemented by Github user kopelli
- Bug Fix: Fixed an exploit that allowed calling any Netscript function without incurring any RAM Cost in NetscriptJS

1.8.71 v0.37.2 - 6/2/2018

- After joining the Bladeburners division, there is now a button to go to the Bladeburner content in the 'City' page
- You now start with \$250m in BitNode-8 (increased from \$100m)
- Bug Fix: You can now no longer directly edit Hacknet Node values through NetscriptJS (hopefully)
- Bug Fix: Bladeburners is no longer accessible in BN-8
- Bug Fix: getBitNodeMultipliers() Netscript function now returns a copy rather than the original object

1.8.72 v0.37.1 - 5/22/2018

- You now earn money from successfully completing Bladeburner contracts. The amount you earn is based on the difficulty of the contract.
- Completing Field Analysis in Bladeburner now grants 0.1 rank
- The maximum RAM you can get on a purchased server is now 1,048,576 GB (2^{20})
- Bug Fix: Fixed Netscript syntax highlighting issues with the new NetscriptJS
- Bug Fix: Netscript Functions now properly incur RAM costs in NetscriptJS
- Bug Fix: deleteServer() now fails if its called on the server you are currently connected to
- Removed in-game Netscript documentation, since it was outdated and difficult to maintain.
- Bug Fix: Updated the gymWorkout() Singularity function with the new exp/cost values for gyms

1.8.73 v0.37.0 - 5/20/2018

- NetscriptJS (Netscript 2.0) released (Documentation here: <http://bitburner.readthedocs.io/en/latest/netscriptjs.html>)
- Running the game with the ‘?noScripts’ query will start the game without loading any of your scripts. This should be used if you accidentally write a script that crashes your game

1.8.74 v0.36.1 - 5/11/2018

- **Bladeburner Changes:**
 - Bug Fix: You can no longer get Bladeburner faction reputation through Infiltration
 - Initial difficulty of Tracking contracts reduced
 - Datamancer skill effect increased from 4% per level to 5%
 - Slightly decreased the base stamina cost of contracts/operations
 - Slightly increased the effects of the Tracer, Digital Observer, Short Circuit, Cloak, and Blade’s Intuition skills
 - Overclock skill capped at level 95, rather than 99
 - Training gives significantly more exp/s
- Crime, Infiltration, and Hacking are now slightly more profitable in BN-6
- Gyms are now more expensive, but give slightly more exp
- Added `getScriptName()` and `getHacknetMultipliers()` Netscript functions (added by Github user hydroflame)
- `getScriptRam()` Netscript function now has default value for the second argument, which is hostname/ip (implemented by Github user hydroflame)
- There is now a soft-cap on stock price, which means it’s no longer possible for the price of a stock to reach insanely-high values
- The `ctrl+b` hotkey in the text editor should now also be triggered by `command+b` on OSX (I don’t have OSX so I can’t confirm if this works)
- Many servers now have additional RAM
- Added an option to disable hotkeys/keyboard shortcuts
- Refactored ‘Active Scripts’ UI page to optimize its performance
- Added a new `.fconf` Terminal setting: `ENABLE_TIMESTAMP`
- ‘Netscript Execution Time’, which can be found in the Options, now has a minimum value of 15ms rather than 25ms
- Bug Fix: Fixed a typo in the Fulcrum Technologies company name (Technologies -> Technologies)
- Bug Fix: `hacknetnodes` keyword should no longer incur RAM cost if its in a comment
- Bug Fix: `disableLog()` now works for the `commitCrime()` Netscript function (fixed by Github user hydroflame)

1.8.75 v0.36.0 - 5/2/2018

- Added BN-6: Bladeburners
- Rebalanced many combat Augmentations so that they are slightly less powerful
- Bug Fix: When faction invites are suppressed, an invitation will no longer load the Faction page

1.8.76 v0.35.2 - 3/26/2018

- **Corporation Changes:**
 - Fixed an issue with Warehouse upgrade cost. Should now be significantly cheaper than before.
 - Scientific Research now has a slightly more significant effect on Product quality
 - The Energy and Water Utilities industries are now slightly more profitable
 - The Robotics and Computer Hardware industries are now less profitable
 - The Software industry is slightly less profitable
 - When selling Materials and Products, the ‘PROD’ qualifier can now be used to set dynamic sell amounts based on your production
 - Exporting MAX should now work properly
 - You can no longer export past storage limits
 - Scientific Research production reduced
 - Effects of AdVert. Inc upgrade were reduced, but the effect that popularity and awareness have on sales was increased to compensate (popularity/awareness numbers were getting too big with Advert. Inc)
 - Bug Fix: Products from Computer Hardware division should now properly have ratings
- Improved Augmentation UI/UX. Now contains collapsible headers and sort buttons
- Improved Faction Augmentations display UI/UX. Now contains sort buttons. There is also an option to disable confirmation when purchasing Augmentations

1.8.77 v0.35.1 - 3/12/2018

- You can now easily download all of your scripts/text files as zip folders. Use the ‘help download’ Terminal command for details
- Scripts are now downloaded with the .script.js extension at the end of their filename
- **Corporation Management Changes:**
 - Implemented Smart Supply unlock
 - Changed the way a division’s Production Multiplier is calculated. It is now the sum of the individual Production Multiplier for every city. Therefore, it is now beneficial to open offices in different cities
 - Several small UI/UX improvements
 - Numerous balance changes. The significant ones are listed below.
 - Product descriptions will now display their estimated market price
 - The sale price of Products can no longer be marked up as high as before

- Scientific Research now affects the rating of Products
 - In general, the maximum amount of product you are able to sell is reduced
 - Sale bonus from advertising (popularity/awareness) now has diminishing returns rather than scaling linearly
- Experience gained during Infiltration now scales linearly based on the clearance level you reach. Compared to before, the experience gained will be much less at lower clearance levels, but much more at higher clearance levels
- The editor can now be used to edit both scripts and text files
- New Terminal config file that can be edited using the command ‘nano .fconf’. Right now there is only one option, but there will be more in the future.
- You can now enable Bash-style Terminal hotkeys using the .fconf file referenced above
- Bug Fix: Fixed an issue with the UI elements of Gang Management persisting across different instances of BitNode-2

1.8.78 v0.35.0 - 3/3/2018

- Minor rebalancing of BitNodes due to the fact that Corporations provide a (relatively) new method of progressing
- **Corporation Management Changes:**
 - Once your Corporation gets big/powerful enough, you can now bribe Factions for reputation using company funds an/or stock shares
 - You can now only create one Division for every Industry type
 - Added several new UI/UX elements
 - Wilson Analytics multiplier was significantly reduced to 1% per level (additive).
 - Reduced the effect of Advert Inc upgrade. Advert Inc. upgrade price increases faster
 - Materials can now be marked up at higher prices
- Added Javascript’s built-in Number object to Netscript
- Added getCharacterInformation(), getCompanyFavor(), and getFactionFavor() Netscript Singularity functions
- Rebalanced Singularity Function RAM Costs. They now cost x8 as much when outside of BN-4 (rather than x10). Also, many of the functions now use significantly less RAM
- Refactored Netscript Ports. You can now get a handle for a Netscript port using the getPortHandle() Netscript function. This allows you to access a port’s underlying queue (which is just an array) and also makes several new functions available such as tryWrite(), full(), and empty().
- Number of Netscript Ports increased from 10 to 20
- Netscript assignments should now return proper values. i.e. `i = 5` should return 5.
- Added throw statements to Netscript. It’s not super useful since ‘catch’ isn’t implemented, but it can be used to generate custom runtime error messages.
- Added import declaration to Netscript. With this, you are able to import functions (and only functions) from other files. Using export declarations is not necessary
- Most Netscript Runtime errors (the ones that cause your script to crash) should now include the line number where the error occurred

- When working for a company, your current company reputation is now displayed
- Whenever you get a Faction Invite it will be immediately appended to your 'invited factions' list. Therefore the checkFactionInvitations() Singularity Function should now be properly useable since you no longer need to decline a Faction Invitation before it shows up in the result.
- Bug Fix: When purchasing servers, whitespace should now automatically be removed from the hostname
- Bug Fix: Can no longer have whitespace in the filename of text files created using write()
- Bug Fix: In Netscript, you can no longer assign a Hacknet Node handle (hacknetnodes[i]) to another value
- Bug Fix: If you are in the Factions tab when you accept an invitation from a Faction, the page will now properly 'refresh'
- Bug Fix: Scripts that run recursive functions should now be killed properly

1.8.79 v0.34.5 - 2/24/2018

- **Corporation Management Changes:**
 - Market Research unlocks are now cheaper
 - New 'VeChain' upgrade: displays useful statistics about Corporation
 - Corporation cycles are processed 25% faster
 - Corporation valuation was lowered by ~10% (this affects stock price and investments)
 - Rebalanced the effects of advertising. Should now be more effective for every Industry
 - Fixed several bugs/exploits involving selling and buying back stock shares
 - You will now receive a Corporation Handbook (.lit file) when starting out BitNode-3. It contains a brief guide to help you get started. This same handbook can be viewed from the Corporation management screen
 - Slightly decreased the amount by which a Product's sell price can be marked up
 - Employees can now be assigned to a 'Training' task, during which they will slowly increase several of their stats
- Hopefully fixed an exploit with Array.forEach(). If there are any issues with using forEach, let me know
- Arguments passed into a script are now passed by value. This means modifying the 'args' array in a script should no longer cause issues
- Scripts executed programatically (via run(), exec(), etc.) will now fail if null/undefined is passed in as an argument
- Added peek() Netscript function
- killall() Netscript function now returns true if any scripts were killed, and false otherwise.
- hack() Netscript function now returns the amount of money gained for successful hacks, and 0 for failed hacks
- scp Terminal command and Netscript function now work for txt files
- **Changes courtesy of Wraithan:**
 - Text files are now displayed using 'pre' rather than 'p' elements when using the 'cat' Terminal command. This means tabs are retained and lines don't automatically wrap
 - ls() Netscript function now returns text files as well
- Removed round() Netscript function, since you can just use Math.round() instead

- Added `disableLog()` and `enableLog()` Netscript functions
- Removed the 'log' argument from `sleep()`, since you can now use the new `disableLog` function
- 'Netscript Documentation' button on script editor now points to new `readthedocs` documentation rather than wiki
- When working for a faction, your current faction reputation is now displayed
- Bug Fix: Hacking Missions should no longer break when dragging an existing connection to another Node
- Bug Fix: Fixed RAM usage of `getNextHacknetNodeCost()` (is not 1.5GB instead of 4GB)

1.8.80 v0.34.4 - 2/14/2018

- Added several new features to Gang UI to make it easier to manage your Gang.
- Changed the Gang Member upgrade mechanic. Now, rather than only being able to have one weapon/armor/vehicle/etc., you can purchase all the upgrades for each Gang member and their multipliers will stack. To balance this out, the effects (AKA multipliers) of each Gang member upgrade were reduced.
- Added a new script editor option: Max Error Count. This affects how many approximate lines the script editor will process (JSHint) for common errors. Increasing this option can affect negatively affect performance
- Game theme colors (set using 'theme' Terminal command) are now saved when re-opening the game
- 'download' Terminal command now works on scripts
- Added `stopAction()` Singularity function and the `spawn()` Netscript function
- The 'Purchase Augmentations' UI screen will now tell you if you need a certain prerequisite for Augmentations.
- Augmentations with prerequisites can now be purchased as long as their prerequisites are purchased (before, you had to actually install the prerequisites before being able to purchase)

1.8.81 v0.34.3 - 1/31/2018

- **Minor balance changes to Corporations:**
 - Upgrades are generally cheaper and/or have more powerful effects.
 - You will receive more funding while your are a private company.
 - Product demand decreases at a slower rate.
 - Production multiplier for Industries (receives for owning real estate/hardware/robots/etc.) is slightly higher
- Accessing the `hacknetnodes` array in Netscript now costs 4.0GB of RAM (only counts against RAM usage once)
- Bug Fix: Corporation outstanding shares should now be numeric rather than a string
- Bug Fix: Corporation production now properly calculated for industries that dont produce materials.
- Bug Fix: Gangs should now properly reset when switching BitNodes
- Bug Fix: Corporation UI should now properly reset when you go public

1.8.82 v0.34.2 - 1/27/2018

- **Corporation Management Changes:**
 - Added advertising mechanics
 - Added Industry-specific purchases
 - Re-designed employee management UI
 - Rebalancing: Made many upgrades/purchases cheaper. Receive more money from investors in early stage. Company valuation is higher after going public
 - Multiple bug fixes
- Added rm() Netscript function
- Updated the way script RAM usage is calculated. Now, a function only increases RAM usage the first time it is called. i.e. even if you call hack() multiple times in a script, it only counts against RAM usage once. The same change applies for while/for loops and if conditionals.
- **The RAM cost of the following were increased:**
 - If statements: increased by 0.05GB
 - run() and exec(): increased by 0.2GB
 - scp(): increased by 0.1GB
 - purchaseServer(): increased by 0.25GB
- Note: You may need to re-save all of your scripts in order to re-calculate their RAM usages. Otherwise, it should automatically be re-calculated when you reset/prestige
- The cost to upgrade your home computer's RAM has been increased (both the base cost and the exponential upgrade multiplier)
- The cost of purchasing a server was increased by 10% (it is now \$55k per RAM)
- Bug fix: (Hopefully) removed an exploit where you could avoid RAM usage for Netscript function calls by assigning functions to a variable (foo = hack(); foo('helios');)
- Bug fix: (Hopefully) removed an exploit where you could run arbitrary Javascript code using the constructor() method
- Thanks to Github user mateon1 and Reddit users havoc_mayhem and spaceglace for notifying me of the above exploits
- The fileExists() Netscript function now works on text files (.txt). Thanks to Github user devoidfury for this

1.8.83 v0.34.1 - 1/19/2018

- **Updates to Corporation Management:**
 - Added a number of upgrades to various aspects of your Corporation
 - Rebalanced the properties of Materials and the formula for determining the valuation of the Corporation
 - Fixed a number of bugs
- 'Stats' page now shows information about current BitNode
- You should now be able to create Corporations in other BitNodes if you have Source-File 3

- Added a new create-able program called `b1t_flum3.exe`. This program can be used to reset and switch BitNodes
- Added an option to adjust autosave interval
- Line feeds, newlines, and tabs will now work with the `tprint()` Netscript function
- Bug fix: 'check' Terminal command was broken
- Bug fix: 'theme' Terminal command was broken when manually specifying hex codes
- Bug fix: Incorrect promotion requirement for 'Business'-type jobs
- Bug fix: Settings input bars were incorrectly formatted when loading game

1.8.84 v0.34.0 - 12/6/2017

- Added `clear()` and `exit()` Netscript functions
- When starting out or prestiging, you will now receive a 'Hacking Starter Guide'. It provides tips/pointers for new players
- Doubled the amount of RAM on low-level servers (up to required hacking level 150)
- Slightly increased experience gain from Infiltration
- `buyStock()`, `sellStock()`, `shortStock()`, and `sellShort()` Netscript function now return the stock price at which the transaction occurred, rather than a boolean. If the function fails for some reason, 0 will be returned.
- **Hacking Mission Changes:**
 - You can now select multiple Nodes of the same type by double clicking. This allows you to set the action of all of selected nodes at once (e.g. set all Transfer Nodes to Fortify). Creating connections does not work with this multi-select functionality yet
 - Shield and Firewall Nodes can now fortify
 - The effects of Fortifying are now ~5% lower
 - Conquering a Spam Node now increases your time limit by 25 seconds instead of 15
 - Damage dealt by Attacking was slightly reduced
 - The effect of Scanning was slightly reduced
 - Enemy CPU Core Nodes start with slightly more attack. Misc Nodes start with slightly less defense
- **Corporation Management changes:**
 - Added several upgrades that unlock new features
 - Implemented Exporting mechanic
 - Fixed many bugs

1.8.85 v0.33.0 - 12/1/2017

- Added BitNode-3: Corporatocracy. In this BitNode you can start and manage your own corporation. This feature is incomplete. Much more will be added to it in the near future
- Minor bug fixes

1.8.86 v0.32.1 - 11/2/2017

- Updated Netscript's 'interpreter/engine' to use the Bluebird promise library instead of native promises. It should now be faster and more memory-efficient. If this has broken any Netscript features please report it through Github or the subreddit (reddit.com/r/bitburner)
- Rebalanced stock market (adjusted parameters such as the volatility/trends/starting price of certain stocks)
- Added prompt() Netscript function
- Added 'Buy Max' and 'Sell All' functions to Stock Market UI
- Added 'Portfolio' Mode to Stock Market UI so you can only view stocks you have a position/order in
- Added a button to kill a script from its log display box

1.8.87 v0.32.0 - 10/25/2017

- Added BitNode-8: Ghost of Wall Street
- Re-designed Stock Market UI
- Minor bug fixes

1.8.88 v0.31.0 - 10/15/2017

- Game now saves to IndexedDb (if your browser supports it). This means you should no longer have trouble saving the game when your save file gets too big (from running too many scripts). The game will still be saved to localStorage as well
- New file type: text files (.txt). You can read or write to text files using the read()/write() Netscript commands. You can view text files in Terminal using 'cat'. Eventually I will make it so you can edit them in the editor but that's not available yet. You can also download files to your real computer using the 'download' Terminal command
- Added a new Crime: Bond Forgery. This crime takes 5 minutes to attempt and gives \$4,500,000 if successful. It is meant for mid game.
- Added commitCrime(), getCrimeChance(), isBusy(), and getStats() Singularity Functions.
- Removed getIntelligence() Netscript function
- Added sprintf and vsprintf to Netscript. See [<https://github.com/alexei/sprintf.js> this Github page for details]
- Increased the amount of money gained from Infiltration by 20%, and the amount of faction reputation by 12%
- Rebalanced BitNode-2 so that Crime and Infiltration are more profitable but hacking is less profitable. Infiltration also gives more faction rep
- Rebalanced BitNode-4 so that hacking is slightly less profitable
- Rebalanced BitNode-5 so that Infiltration is more profitable and gives more faction rep
- Rebalanced BitNode-11 so that Crime and Infiltration are more profitable. Infiltration also gives more faction rep.
- Fixed an annoying issue in Hacking Missions where sometimes you would click a Node but it wouldn't actually get selected
- Made the Hacking Mission gameplay a bit slower by lowering the effect of Scan and reducing Attack damage
- Slightly increased the base reputation gain rate for factions when doing Field Work and Security Work

1.8.89 v0.30.0 - 10/9/2017

- Added getAugmentations() and getAugmentationsFromFaction() Netscript Singularity Functions
- Increased the rate of Intelligence exp gain
- Added a new upgrade for home computers: CPU Cores. Each CPU core on the home computer grants an additional starting Core Node in Hacking Missions. I may add in other benefits later. Like RAM upgrades, upgrading the CPU Core on your home computer persists until you enter a new BitNode.
- Added lscpu Terminal command to check number of CPU Cores
- Changed the effect of Source-File 11 and made BitNode-11 a little bit harder
- Fixed a bug with Netscript functions (the ones you create yourself)
- **Hacking Missions officially released (they give reputation now). Notable changes in the last few updates:**
 - Misc Nodes slowly gain hp/defense over time
 - Conquering a Misc Node will increase the defense of all remaining Misc Nodes that are not being targeted by a certain percentage
 - Reputation reward for winning a Mission is now affected by faction favor and Player's faction rep multiplier
 - Whenever a Node is conquered, its stats are reduced

1.8.90 v0.29.3 - 10/3/2017

- Fixed bug for killing scripts and showing error messages when there are errors in a player-defined function
- Added function name autocompletion in Script Editor. Press Ctrl+space on a prefix to show autocompletion options.
- Minor rebalancing and bug fixes for Infiltration and Hacking Missions

1.8.91 v0.29.2 - 10/1/2017

- installAugmentations() Singularity Function now takes a callback script as an argument. This is a script that gets ran automatically after Augmentations are installed. The script is run with no arguments and only a single thread, and must be found on your home computer.
- Added the ability to create your own functions in Netscript. See [\[Netscript Functions|this link\]](#) for details
- Added :q, :x, and :wq Vim Ex Commands when using the Vim script editor keybindings. :w, :x, and :wq will all save the script and return to Terminal. :q will quit (return to Terminal) WITHOUT saving. If anyone thinks theres an issue with this please let me know, I don't use Vim
- Added a new Augmentation: ADR-V2 Pheromone Gene
- In Hacking Missions, enemy nodes will now automatically target Nodes and perform actions.
- Re-balanced Hacking Missions through minor tweaking of many numbers
- The faction reputation reward for Hacking Missions was slightly increased

1.8.92 v0.29.1 - 9/27/2017

- New gameplay feature that is currently in BETA: Hacking Missions. Hacking Missions is an active gameplay mechanic (its a minigame) that is meant to be used to earn faction reputation. However, since this is currently in beta, hacking missions will NOT grant reputation for the time being, since the feature likely has many bugs, balance problems, and other issues. If you have any feedback regarding the new feature, feel free to let me know
- CHANGED THE RETURN VALUE OF `getScriptIncome()` WHEN RAN WITH NO ARGUMENTS. It will now return an array of two values rather than a single value. This may break your scripts, so make sure to update them!
- Added `continue` statement for `for/while` loops
- Added `getServerMinSecurityLevel()`, `getPurchasedServers()`, and `getTimeSinceLastAug()` Netscript functions
- Netscript `scp()` function can now take an array as the first argument, and will try to copy every file specified in the array (it will just call `scp()` normally for every element in the array). If an array is passed in, then the `scp()` function returns `true` if at least one element from the array is successfully copied
- Added Javascript's `Date` module to Netscript. Since 'new' is not supported in Netscript yet, only the `Date` module's static methods will work (`now()`, `UTC()`, `parse()`, etc.).
- Failing a crime now gives half the experience it did before
- The forced repeated 'Find The-Cave' message after installing The Red Pill Augmentation now only happens if you've never destroyed a BitNode before, and will only popup every 15 minutes. If you have already destroyed a BitNode, the message will not pop up if you have messages suppressed (if you don't have messages suppressed it WILL still repeatedly popup)
- `fileExists()` function now works on literature files

1.8.93 v0.29.0 - 9/19/2017

- Added BitNode-5: Artificial Intelligence
- Added `getIp()`, `getIntelligence()`, `getHackingMultipliers()`, and `getBitNodeMultipliers()` Netscript functions (requires Source-File 5)
- Updated `scan()` Netscript function so that you can choose to have it print IPs rather than hostnames
- Refactored `scp()` Netscript function so that it takes an optional 'source server' argument
- For Infiltration, decreased the percentage by which the security level increases by about 10% for every location
- Using `:w` in the script editor's Vim keybinding mode should now save and quit to Terminal
- Some minor optimizations that should reduce the size of the save file
- `scan-analyze` Terminal command will no longer show your purchased servers, unless you pass a '-a' flag into the command
- After installing the Red Pill augmentation from Daedalus, the message telling you to find 'The-Cave' will now repeatedly pop up regardless of whether or not you have messages suppressed
- Various bugfixes

1.8.94 v0.28.6 - 9/15/2017

- Time required to create programs now scales better with hacking level, and should generally be much faster
- Added `serverExists(hostname/ip)` and `getScriptExpGain(scriptname, ip, args...)` Netscript functions

- Short circuiting && and || logical operators should now work
- Assigning to multidimensional arrays should now work
- Scripts will no longer wait for hack/grow/weaken functions to finish if they are killed. They will die immediately
- The script loop that checks whether any scripts need to be started/stopped now runs every 6 seconds rather than 10 (resulting in less delays when stopping/starting scripts)
- Fixed several bugs/exploits
- Added some description for BitNode-5 (not implemented yet, should be soon though)

1.8.95 v0.28.5 - 9/13/2017

- The fl1ght.exe program that is received from jump3r is now sent very early on in the game, rather than at hacking level 1000
- Hostname is now displayed in Terminal
- Syntax highlighting now works for all Netscript functions
- Export should now work on Edge/IE

1.8.96 v0.28.4 - 9/11/2017

- Added getScriptIncome() Netscript function
- Added Javascript's math module to Netscript. See [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math this link for details]
- Added several member variables for the Hacknet Node API that allow you to access info about their income
- All valid Netscript functions are now syntax highlighted as keywords in the editor. This means they will a different color than invalid netscript functions. The color will depend on your theme. Note that right now, this only applies for normal Netscript functions, not functions in the TIX API, Hacknet Node API, or Singularity Functions.
- Comments and operators no longer count towards RAM usage in scripts.
- Variety of bug fixes and updates to informational text in the game

1.8.97 v0.28.3 - 9/7/2017

- Added ls() Netscript function
- Increased company wages by about ~10% across the board
- The scp() Netsection function and Terminal command now works for .lit files
- Increased the amount of RAM on many lower level servers (up to level 200 hacking level required).

1.8.98 v0.28.2 - 9/4/2017

- Added several configuration options for script editor (key bindings, themes, etc.)
- Certain menu options will now be hidden until their relevant gameplay is unlocked. This includes the Factions, Augmentations, Create Program, Travel, and Job tabs. This will only affect newer players.

- Most unrecognized or un-implemented syntax errors in Netscript will now include the line number in the error message

1.8.99 v0.28.1 - 9/1/2017

- The script editor now uses the open-source Ace editor, which provides a much better experience when coding!
- Added tprint() Netscript function

1.8.100 v0.28.0 - 8/30/2017

- Added BitNode-4: The Singularity
- Added BitNode-11: The Big Crash
- Migrated the codebase to use webpack (doesn't affect any in game content, except maybe some slight performance improvements and there may be bugs that result from dependency errors)

1.8.101 v0.27.3 - 8/19/2017

- You can now purchase upgrades for Gang Members (BitNode 2 only)
- Decreased Gang respect gains and slightly increased wanted gains (BitNode 2 only)
- Other gangs will increase in power faster (BitNode 2 only)
- Added getHackTime(), getGrowTime(), and getWeakenTime() Netscript functions

1.8.102 v0.27.2 - 8/18/2017

- Added getServerGrowth() Netscript function
- Added getNextHacknetNodeCost() Netscript function
- Added new 'literature' files (.lit extension) that are used to build lore for the game. These .lit files can be found in certain servers throughout the game. They can be viewed with the 'cat' Terminal command and copied over to other servers using the 'scp' command. These .lit files won't be found until you reset by installing Augmentations
- Fixed some bugs with Gang Territory(BitNode 2 only)

1.8.103 v0.27.1 - 8/15/2017

- Changed the way Gang power was calculated to make it scale better late game (BitNode 2 only)
- Lowered the respect gain rate in Gangs (Bitnode 2 only)
- Added 'l grep pattern' option for ls Terminal command. This allows you to only list files that contain a certain pattern
- Added break statement in Netscript
- Display for some numerical values is now done in shorthand (e.g 1.000m instead of 1,000,000)

1.8.104 v0.27.0 - 8/13/2017

- Added secondary ‘prestige’ system - featuring Source Files and BitNodes
- MILD SPOILERS HERE: Installing ‘The Red Pill’ Augmentation from Daedalus will unlock a special server called w0rld_d43m0n. Finding and manually hacking this server through Terminal will destroy the Player’s current BitNode, and allow the player to enter a new one. When destroying a BitNode, the player loses everything except the scripts on his/her home computer. The player will then gain a powerful second-tier persistent upgrade called a Source File. The player can then enter a new BitNode to start the game over. Each BitNode has different characteristics, and many will have new content/mechanics as well. Right now there are only 2 BitNodes. Each BitNode grants its own unique Source File. Restarting and destroying a BitNode you already have a Source File for will upgrade your Source File up to a maximum level of 3.
- Reputation gain with factions and companies is no longer a linear conversion, but an exponential one. It will be much easier to gain faction favor at first, but much harder later on.
- Significantly increased Infiltration exp gains
- Fixed a bug with company job requirement tooltips
- Added scriptRunning(), scriptKill(), and getScriptRam() Netscript functions. See documentation for details
- Fixed a bug with deleteServer() Netscript function

1.8.105 v0.26.4 - 8/1/2017

- All of the ‘low-level servers’ in early game that have a required hacking level now have 8GB of RAM instead of 4GB
- Increased the amount of experience given at university
- Slightly increased the production of Hacknet Nodes and made them cheaper to upgrade
- Infiltration now gives slightly more EXP and faction reputation
- Added two new crimes. These crimes are viable to attempt early on in the game and are relatively passive (each take 60+ seconds to complete)
- Crimes give more exp and more money
- Max money available on a server decreased from 50x the server’s starting money to 25x
- Significantly increased wages for all jobs

1.8.106 v0.26.3

- Added support for large numbers using Decimal.js. Right now it only applies for the player’s money
- Purchasing servers with the Netscript function purchaseServer() is no longer 2x as expensive as doing manually it now costs the same
- Early game servers have more starting money

1.8.107 v0.26.2

- Major rebalancing and randomization of the amount of money that servers start with
- Significantly lowered hacking exp gain from hacking servers. The exp gain for higher-level servers was lowered more than that of low level servers. (~16% for lower level servers, up to ~25% for higher-level servers)

- Added deleteServer() Netscript function
- You can now purchase a maximum of 25 servers each run (Deleting a server will allow you to purchase a new one)
- Added autocompletion for './' Terminal command
- Darkweb prices now displayed properly using toLocaleString()
- Added NOT operator (!) and negation operator(-) in Netscript, so negative numbers should be functional now
- Rejected faction invitations will now show up as 'Outstanding Faction Invites' in the Factions page. These can be accepted at any point in the future
- Added a few more configurable game settings for suppressing messages and faction invitations
- Added tooltips for company job requirements

1.8.108 v0.26.1

- Added autocompletion for aliases
- Added getServerRam() Netscript function()
- Added getLevelUpgradeCost(n), getRamUpgradeCost(), getCoreUpgradeCost() functions for Netscript Hacknet Node API
- Added some configurable settings (See Game Options menu)

1.8.109 v0.26.0

- Game now has a real ending, although it's not very interesting/satisfying right now. It sets up the framework for the secondary prestige system in the future
- Forgot to mention that since last update, comments now work in Netscript. Use // for single line comments or /* and */ for multiline comments just like in Javascript
- Added ports to Netscript. These ports are essentially serialized queues. You can use the write() Netscript function to write a value to a queue, and then you can use the read() Netscript function to read the value from the queue. Once you read a value from the queue it will be removed. There are only 10 queues (1-10), and each has a maximum capacity of 50 entries. If you try to write to a queue that is full, the the first value is removed. See wiki/Netscript documentation for more details
- You can now use the 'help' Terminal command for specific commands
- You can now use './' to run a script/program (./NUKE.exe). However, tab completion currently doesn't work for it (I'm working on it)
- Decreased the base growth rate of servers by ~25%
- Both the effect of weaken() and its time to execute were halved. In other words, calling weaken() on a server only lowers its security by 0.05 (was 0.1 before) but the time to execute the function is half of what it was before. Therefore, the effective rate of weaken() should be about the same
- Increased all Infiltration rewards by ~10%, and increased infiltration rep gains by an additional 20% (~32% total for rep gains)
- The rate at which the security level of a facility increases during Infiltration was decreased significantly (~33%)
- Getting treated at the Hospital is now 33% more expensive
- Slightly increased the amount of time it takes to hack a server

- Slightly decreased the amount of money gained when hacking a server (~6%)
- Slightly decreased the base cost for RAM on home computer, but increased the cost multiplier. This means that upgrading RAM on the home computer should be slightly cheaper at the start, but slightly more expensive later on
- Increased the required hacking level for many late game servers
- The sleep() Netscript function now takes an optional 'log' argument that specifies whether or not the 'Sleeping for N milliseconds' will be logged for the script
- Added clearLog() Netscript function
- Deleted a few stocks. Didn't see a reason for having so many, and it just affects performance. Won't take effect until you reset by installing Augmentations
- There was a typo with Zeus Medical's server hostname. It is now 'zeus-med' rather than 'zeud-med'
- Added keyboard shortcuts to quickly navigate between different menus. See wiki link (<http://bitburner.wikia.com/wiki/Shortcuts>)
- Changed the Navigation Menu UI

1.8.110 v0.25.0

- Refactored Netscript to use the open-source Acorns Parser. This re-implementation was done by [<https://github.com/MrNuggelz> Github user MrNuggelz]. This has resulted in several changes in the Netscript language. Some scripts might break because of these changes. Changes listed below:
- Arrays are now fully functional Javascript arrays. You no longer need to use the 'Array' keyword to declare them.
- The length(), clear/clear(), insert(), and remove() functions no longer work for arrays.
- All Javascript array methods are available (splice(), push(), pop(), join(), shift(), indexOf(), etc. See documentation)
- Variables assigned to arrays are now passed by value rather than reference
- Incrementing/Decrementing are now available (i++, ++i)
- You no longer need semicolons at the end of block statements
- Elif is no longer valid. Use 'else if' instead
- Netscript's Hacknet Node API functions no longer log anything
- Stock prices now update every ~6 seconds when the game is active (was 10 seconds before)
- Added a new mechanic that affects how stock prices change
- Script editor now has dynamic indicators for RAM Usage and Line number
- Augmentation Rebalancing - Many late game augmentations are now slightly more expensive. Several early game augmentations had their effects slightly decreased
- Increased the amount of rewards (both money and rep) you get from infiltration
- Purchasing servers is now slightly more expensive
- Calling the Netscript function getServerMoneyAvailable('home') now return's the player's money
- Added round(n) Netscript function - Rounds a number
- Added purchaseServer(hostname, ram) Netscript function

- Added the TIX API. This must be purchased in the WSE. It persists through resets. Access to the TIX API allows you to write scripts that perform automated algorithmic trading. See Netscript documentation
- Minor rebalancing in a lot of different areas
- Changed the format of IP Addresses so that they are smaller (will consist mostly of single digit numbers now). This will reduce the size of the game's save file.

1.8.111 v0.24.1

- Adjusted cost of upgrading home computer RAM. Should be a little cheaper for the first few upgrades (up to ~64GB), and then will start being more expensive than before. High RAM upgrades should now be significantly more expensive than before.
- Slightly lowered the starting money available on most mid-game and end-game servers (servers with required hacking level greater than 200) by about 10-15%
- Rebalanced company/company position reputation gains and requirements
- Studying at a university now gives slightly more EXP and early jobs give slightly less EXP
- Studying at a university is now considerably more expensive
- Rebalanced stock market
- Significantly increased cost multiplier for purchasing additional Hacknet Nodes
- The rate at which facility security level increases during infiltration for each clearance level was lowered slightly for all companies
- Updated Faction descriptions
- Changed the way alias works. Normal aliases now only work at the start of a Terminal command (they will only replace the first word in the Terminal command). You can also create global aliases that work on any part of the command, like before. Declare global aliases by entering the optional -g flag: `alias -g name="value"` - [<https://github.com/MrNuggelz> Courtesy of Github user MrNuggelz]
- 'top' Terminal command implemented courtesy of [<https://github.com/LTCNugget> Github user LTCNugget]. Currently, the formatting gets screwed up if your script names are really long.

1.8.112 v0.24.0

- Players now have HP, which is displayed in the top right. To regain HP, visit the hospital. Currently the only way to lose HP is through infiltration
- Infiltration - Attempt to infiltrate a company and steal their classified secrets. See 'Companies' documentation for more details
- Stock Market - Added the World Stock Exchange (WSE), a brokerage that lets you buy/sell stocks. To begin trading you must first purchase an account. A WSE account will persist even after resetting by installing Augmentations. How the stock market works should hopefully be self explanatory. There is no documentation about it currently, I will add some later. NOTE: Stock prices only change when the game is open. The Stock Market is reset when installing Augmentations, which means you will lose all your stocks
- Decreased money gained from hacking by ~12%
- Increased reputation required for all Augmentations by ~40%
- Cost increase when purchasing multiple augmentations increased from 75% to 90%

- Added basic variable runtime to Netscript operations. Basic commands run in 100ms. Any function incurs another 100ms in runtime (200ms total). Any function that starts with `getServer` incurs another 100ms runtime (300ms total). `exec()` and `scp()` require 400ms total.
- Slightly reduced the amount of experience gained from hacking

1.8.113 v0.23.1

- `scan()` Netscript function now takes a single argument representing the server from which to scan.

1.8.114 v0.23.0

- You can now purchase multiple Augmentations in a run. When you purchase an Augmentation you will lose money equal to the price and then the cost of purchasing another Augmentation during this run will be increased by 75%. You do not gain the benefits of your purchased Augmentations until you install them. This installation can be done through the 'Augmentation' tab. When you install your Augmentations, your game will reset like before.
- Reputation needed to gain a favor from faction decreased from 7500 to 6500
- Reputation needed to gain a favor from company increased from 5000 to 6000
- Reputation cost of all Augmentations increased by 16%
- Higher positions at companies now grant slightly more reputation for working
- Added `getServerMaxMoney()` Netscript function
- Added `scan()` Netscript function
- Added `getServerNumPortsRequired()` Netscript function
- There is now no additional RAM cost incurred when multithreading a script

1.8.115 v0.22.1

- You no longer lose progress on creating programs when cancelling your work. Your progress will be saved and you will pick up where you left off when you start working on it again
- Added two new programs: `AutoLink.exe` and `ServerProfiler.exe`
- Fixed bug with Faction Field work reputation gain

1.8.116 v0.22.0 - Major rebalancing, optimization, and favor system

- Significantly nerfed most augmentations
- Almost every server with a required hacking level of 200 or more now has slightly randomized server parameters. This means that after every Augmentation purchase, the required hacking level, base security level, and growth factor of these servers will all be slightly different
- The hacking speed multiplier now increases rather than decreases. The hacking time is now divided by your hacking speed multiplier rather than multiplied. In other words, a higher hacking speed multiplier is better
- Servers now have a minimum server security, which is approximately one third of their starting ('base') server security

- If you do not steal any money from a server, then you gain hacking experience equal to the amount you would have gained had you failed the hack
- The effects of grow() were increased by 50%
- grow() and weaken() now give hacking experience based on the server's base security level, rather than a flat exp amount
- Slightly reduced amount of exp gained from hack(), weaken(), and grow()
- Rebalanced formulas that determine crime success
- Reduced RAM cost for multithreading a script. The RAM multiplier for each thread was reduced from 1.02 to 1.005
- Optimized Script objects so they take less space in the save file
- Added getServerBaseSecurityLevel() Netscript function
- New favor system for companies and factions. Earning reputation at a company/faction will give you favor for that entity when you reset after installing an Augmentation. This favor persists through the rest of the game. The more favor you have, the faster you will earn reputation with that faction/company
- You can no longer donate to a faction for reputation until you have 150 favor with that faction
- Added unalias Terminal command
- Changed requirements for endgame Factions

1.8.117 v0.21.1

- IF YOUR GAME BREAKS, DO THE FOLLOWING: Options -> Soft Reset -> Save Game -> Reload Page. Sorry about that!
- Autocompletion for aliases - courtesy of [<https://github.com/LTCNugget> Github user LTCNugget]

1.8.118 v0.21.0

- Added dynamic arrays. See Netscript documentation
- Added ability to pass arguments into scripts. See documentation
- The implementation/function signature of functions that deal with scripts have changed. Therefore, some old scripts might not work anymore. Some of these functions include run(), exec(), isRunning(), kill(), and some others I may have forgot about. Please check the updated Netscript documentation if you run into issues.-Note that scripts are now uniquely identified by the script name and their arguments. For example, you can run a script using:

```
run foodnstuff.script 1
```

and you can also run the same script with a different argument:

```
run foodnstuff.script 2
```

These will be considered two different scripts. To kill the first script you must run:

```
kill foodnstuff.script 1
```

and to kill the second you must run:

```
kill foodnstuff.script 2
```

Similar concepts apply for Terminal Commands such as `tail`, and Netscript commands such as `run()`, `exec()`, `kill()`, `isRunning()`, etc.

- Added basic theme functionality using the ‘theme’ Terminal command - All credit goes to /u/0x726564646974 who implemented the awesome feature
- Optimized Script objects, which were causing save errors when the player had too many scripts
- Formula for determining exp gained from hacking was changed
- Fixed bug where you could purchase Darkweb items without TOR router
- Slightly increased cost multiplier for Home Computer RAM
- Fixed bug where you could hack too much money from a server (and bring its money available below zero)
- Changed tail command so that it brings up a display box with dynamic log contents. To get old functionality where the logs are printed to the Terminal, use the new ‘check’ command
- As a result of the change above, you can no longer call tail/check on scripts that are not running
- Added autocompletion for buying Programs in Darkweb

1.8.119 v0.20.2

- Fixed several small bugs
- Added basic array functionality to Netscript
- Added ability to run scripts with multiple threads. Running a script with `n` threads will multiply the effects of all `hack()`, `grow()`, and `weaken()` commands by `n`. However, running a script with multiple threads has drawbacks in terms of RAM usage. A script’s ram usage when it is ‘multithreaded’ is calculated as: $\text{base cost} * \text{numThreads} * (1.02 ^ \text{numThreads})$. A script can be run multithreaded using the ‘run [script] -t n’ Terminal command or by passing in an argument to the `run()` and `exec()` Netscript commands. See documentation.
- RAM is slightly (~10%) more expensive (affects purchasing server and upgrading RAM on home computer)
- NeuroFlux Governor augmentation cost multiplier decreased
- Netscript default operation runtime lowered to 200ms (was 500ms previously)

1.8.120 v0.20.1

- Fixed bug where sometimes scripts would crash without showing the error
- Added Deepscan programs to Dark Web
- Declining a faction invite will stop you from receiving invitations from that faction for the rest of the run
- (BETA) Added functionality to export/import saves. WARNING This is only lightly tested. You cannot choose where to save your file it just goes to the default save location. Also I have no idea what will happen if you try to import a file that is not a valid save. I will address these in later updates

1.8.121 v0.20.0

- Refactored Netscript Interpreter code. Operations in Netscript should now run significantly faster (Every operation such as a variable assignment, a function call, a binary operator, getting a variable's value, etc. used to take up to several seconds, now each one should only take ~500 milliseconds).
- Percentage money stolen when hacking lowered to compensate for faster script speeds
- Hacking experience granted by grow() halved
- Weaken() is now ~11% faster, but only grants 3 base hacking exp upon completion instead of 5
- Rebalancing of script RAM costs. Base RAM Cost for a script increased from 1GB to 1.5GB. Loops, hack(), grow() and weaken() all cost slightly less RAM than before
- Added `getServerRequiredHackingLevel(server)` Netscript command.
- Added `fileExists(file, [server])` Netscript command, which is used to check if a script/program exists on a specified server
- Added `isRunning(script, [server])` Netscript command, which is used to check if a script is running on the specified server
- Added `killall` Terminal command. Kills all running scripts on the current machine
- Added `kill()` and `killall()` Netscript commands. Used to kill scripts on specified machines. See Netscript documentation
- Re-designed 'Active Scripts' tab
- Hacknet Node base production rate lowered from 1.6 to 1.55 (\$/second)
- Increased monetary cost of RAM (Upgrading home computer and purchasing servers will now be more expensive)
- NEW GROWTH MECHANICS - The rate of growth on a server now depends on a server's security level. A higher security level will result in lower growth on a server when using the `grow()` command. Furthermore, calling `grow()` on a server raises that server's security level by 0.004. For reference, if a server has a security level of 10 it will have approximately the same growth rate as before.
- Server growth no longer happens naturally
- Servers now have a maximum limit to their money. This limit is 50 times it's starting money
- Hacking now grants 10% less hacking experience
- You can now edit scripts that are running
- Augmentations cost ~11% more money and 25% more faction reputation

1.8.122 v0.19.7

- Added changelog to Options menu
- Bug fix with autocompletion (wasn't working properly for capitalized filenames/programs)

1.8.123 v0.19.6

- Script editor now saves its state even when you change tabs
- `scp()` command in Terminal/script will now overwrite files at the destination

- Terminal commands are no longer case-sensitive (only the commands themselves such as 'run' or 'nano'. File-names are still case sensitive)
- Tab automcompletion will now work on commands

1.8.124 v0.19.0

- Hacknet Nodes have slightly higher base production, and slightly increased RAM multiplier. But they are also a bit more expensive at higher levels
- Calling grow() and weaken() in a script will now work offline, at slower rates than while online (The script now keeps track of the rate at which grow() and weaken() are called when the game is open. These calculated rates are used to determine how many times the calls would be made while the game is offline)
- Augmentations now cost 20% more reputation and 50% more money
- Changed the mechanic for getting invited to the hacking factions (CyberSec, NiteSec, The Black Hand, BitRunners) Now when you get to the required level to join these factions you will get a message giving you instructions on what to do in order to get invited.
- Added a bit of backstory/plot into the game. It's not fully fleshed out yet but it will be used in the future
- Made the effects of many Augmentations slightly more powerful
- Slightly increased company job wages across the board (~5-10% for each position)
- Gyms and classes are now significantly more expensive
- Doubled the amount by which a server's security increases when it is hacked. Now, it will increase by 0.002. Calling weaken() on a server will lower the security by 0.1.

1.8.125 v0.18.0

- Major rebalancing (sorry didn't record specifics. But in general hacking gives more money and hacknet nodes give less)
- Server growth rate (both natural and manual using grow()) doubled
- Added option to Soft Reset
- Cancelling a full time job early now only results in halved gains for reputation. Exp and money earnings are gained in full
- Added exec() Netscript command, used to run scripts on other servers.
- NEW HACKING MECHANICS: Whenever a server is hacked, its 'security level' is increased by a very small amount. The security level is denoted by a number between 1-100. A higher security level makes it harder to hack a server and also decreases the amount of money you steal from it. Two Netscript functions, weaken() and getServerSecurityLevel() level, were added. The weaken(server) function lowers a server's security level. See the Netscript documentation for more details
- When donating to factions, the base rate is now \$1,000,000 for 1 reputation point. Before, it was \$1,000 for 1 reputation point.
- Monetary costs for all Augmentations increased. They are now about ~3.3 - 3.75 times more expensive than before

1.8.126 v0.17.1

- Fixed issue with purchasing Augmentations that are ‘upgrades’ and require previous Augmentations to be installed
- Increased the percentage of money stolen from servers when hacking

1.8.127 v0.17.0

- Greatly increased amount of money gained for crimes (by about 400% for most crimes)
- Criminal factions require slightly less negative karma to get invited to
- Increased the percentage of money stolen from servers when hacking
- Increased the starting amount of money available on beginning servers (servers with <50 required hacking))
- Increased the growth rate of servers (both naturally and manually when using the grow() command in a script)
- Added getHostname() command in Netscript that returns the hostname of the server a script is running on
- jQuery preventDefault() called when pressing ctrl+b in script editor
- The Neuroflux Governor augmentation (the one that can be repeatedly leveled up) now increases ALL multipliers by 1%. To balance it out, it's price multiplier when it levels up was increased
- Hacknet Node base production decreased from \$1.75/s to \$1.65/s
- Fixed issue with nested for loops in Netscript (stupid Javascript references)
- Added ‘scp’ command to Terminal and Netscript
- Slightly nerfed Hacknet Node Kernel DNI and Hacknet Node Core DNI Augmentations
- Increased TOR Router cost to \$200k

1.8.128 v0.16.0

- New Script Editor interface
- Rebalanced hacknet node - Increased base production but halved the multiplier from additional cores. This should boost its early-game production but nerf its late-game production
- Player now starts with 8GB of RAM on home computer
- ‘scan-analyze’ terminal command displays RAM on servers
- Slightly buffed the amount of money the player steals when hacking servers (by about ~8%)
- Time to execute grow() now depends on hacking skill and server security, rather than taking a flat 2 minutes.
- Clicking outside of a pop-up dialog box will now close it
- BruteSSH.exe takes 33% less time to create
- ‘iron-gym’ and ‘max-hardware’ servers now have 2GB of RAM
- Buffed job salaries across the board
- Updated Tutorial
- Created a Hacknet Node API for Netscript that allows you to access and upgrade your Hacknet Nodes. See the Netscript documentation for more details. WARNING The old upgradeHacknetNode() and getNumHacknetNodes() functions were removed so any script that has these will no longer work

1.8.129 v0.15.0

- Slightly reduced production multiplier for Hacknet Node RAM
- Faction pages now scroll
- Slightly increased amount of money gained from hacking
- Added 'alias' command
- Added 'scan-analyze' terminal command - used to get basic hacking info about all immediate network connections
- Fixed bugs with upgradeHacknetNode() and purchaseHacknetNode() commands
- Added getNumHacknetNodes() and hasRootAccess(hostname/ip) commands to Netscript
- Increased Cost of university classes/gym
- You can now see what an Augmentation does and its price even while its locked

1.9 v1.0.0 Migration Guide

In v1.0.0 a few API have been broken.

migrated (only for ns2):

- bladeburner.getActionTime will return milliseconds instead of seconds.
- getHackTime will return milliseconds instead of seconds.
- getGrowTime will return milliseconds instead of seconds.
- getWeakenTime will return milliseconds instead of seconds.
- hackAnalyzePercent renamed to hackAnalyze
- hackAnalyzePercent will return decimal instead of percentage
- hackChance (not formulas.basic.hackChance) renamed to hackAnalyzeChance
- formulas.basic is split into formulas.skills and formulas.hacking

not migrated (require manual changes sometimes):

- getPlayer().hacking_skill renamed *hacking*
- same thing in sleeves
- getPurchasedServers won't let you query for ips instead of hostnames.
- getStats is deprecated in favor getPlayer
- getCharacterInformation is deprecated in favor getPlayer
- getServerRam deprecated in favor of getServerMaxRam and getServerUsedRam
- getServerBaseSecurityLevel will be deprecated in favor of nothing, it's not really used.
- sleep can no longer be called simultenaously, a new function called asleep will let you.
- write returns promise (needs to be await ed).
- scp returns a promise (needs to be await ed).
- free port, write, read

- write, read does not support port anymore, writePort and readPort does.

Upon loading v1.0.0 the game will apply some rules to change everything. The game never changes a file before making a backup called *BACKUP_filename.ext*, then, in the script it will change whatever it thinks it should change. But will prefix the modified line with the original line.

A file called *v1_DETECTED_CHANGES.txt* will point out every file with some possible problem.

1.10 Hi there, hello

It looks like you found a page that doesn't exist!

If you're looking for documentation of the netscript API. It moved [here](#).

That documentation is autogenerated and therefore is much easier to maintain.

CHAPTER 2

Indices and tables

- `genindex`
- `modindex`
- `search`

A

autocomplete() *(built-in function)*, 27

B

brutessh() *(built-in function)*, 13

D

deleteServer() *(built-in function)*, 25

disableLog() *(built-in function)*, 12

E

enableLog() *(built-in function)*, 12

exec() *(built-in function)*, 15

F

fileExists() *(built-in function)*, 23

ftpcrack() *(built-in function)*, 13

G

getCoreUpgradeCost() *(built-in function)*, 31

getHackingLevel() *(built-in function)*, 19

getHackingMultipliers() *(built-in function)*, 19

getHacknetMultipliers() *(built-in function)*, 20

getLevelUpgradeCost() *(built-in function)*, 31

getNodeStats() *(built-in function)*, 29

getPurchasedServerCost() *(built-in function)*,
24

getPurchasedServerLimit() *(built-in function)*,
25

getPurchasedServerMaxRam() *(built-in function)*, 26

getPurchasedServers() *(built-in function)*, 25

getPurchaseNodeCost() *(built-in function)*, 29

getRamUpgradeCost() *(built-in function)*, 31

getScriptRam() *(built-in function)*, 27

getServerMaxMoney() *(built-in function)*, 21

getServerMaxRam() *(built-in function)*, 22

getServerMinSecurityLevel() *(built-in function)*, 21

getServerMoneyAvailable() *(built-in function)*,
20

getServerNumPortsRequired() *(built-in function)*, 22

getServerRequiredHackingLevel() *(built-in function)*, 21

getServerSecurityLevel() *(built-in function)*,
21

getServerUsedRam() *(built-in function)*, 22

grow() *(built-in function)*, 10

H

hack() *(built-in function)*, 9

hasRootAccess() *(built-in function)*, 19

httpworm() *(built-in function)*, 14

I

isLogEnabled() *(built-in function)*, 12

isRunning() *(built-in function)*, 23, 24

K

kill() *(built-in function)*, 16, 17

killall() *(built-in function)*, 17

L

ls() *(built-in function)*, 18

N

nuke() *(built-in function)*, 13

numNodes() *(built-in function)*, 29

P

print() *(built-in function)*, 11

ps() *(built-in function)*, 18

purchaseNode() *(built-in function)*, 29

purchaseServer() *(built-in function)*, 24

R

relaysmtp() *(built-in function)*, 13

`run()` (*built-in function*), 14

S

`scan()` (*built-in function*), 12

`scp()` (*built-in function*), 17

`scriptKill()` (*built-in function*), 26

`scriptRunning()` (*built-in function*), 26

`serverExists()` (*built-in function*), 22

`sleep()` (*built-in function*), 11

`spawn()` (*built-in function*), 16

`sqlinject()` (*built-in function*), 14

T

`tprint()` (*built-in function*), 11

U

`upgradeCore()` (*built-in function*), 30

`upgradeLevel()` (*built-in function*), 30

`upgradeRam()` (*built-in function*), 30

W

`weaken()` (*built-in function*), 10